commodore

comments and bulletins
concerning your
COMMODORE PET

The Transactor

Vol 2
BULLETIN # 7
Dec. 31, 79

PET[tm] is a registered trademark of Commodore Inc.

This month's Transactor is a collection of all the charts and tables concerning PET and computers in general. Some have appeared in previous Transactors but flipping and finding can be a chore. Therefore a handy reference was thought to be in order.

Also in this Transactor is Bill MacLean's Block Get routine mentioned in Transactor #6 but accidentally omitted

```
10 DATA 120 ,  56 , 169 , 180 , 237
20 DATA 144 ,   0 , 141 , 144 ,   0
30 DATA  56 , 169 , 233 , 237 , 145
40 DATA   0 , 141 , 145 ,   0 ,  88
50 DATA  96 , 173 , 166 ,   0 , 201
60 DATA 255 , 208 ,  12 , 169 ,   0
70 DATA 141 , 103 ,   3 , 169 ,  90
80 DATA 141 , 120 ,   3 , 208 ,  25
90 DATA 238 , 103 ,   3 , 173 , 104
92 DATA   3 , 205 , 103 ,   3 , 176
94 DATA  14 , 169 ,   6 , 141 , 104
96 DATA   3 , 162 , 255 , 142 , 151
98 DATA   0 , 232 , 142 , 103 ,   3
99 DATA  76 ,  46 , 230
100 FOR I=880 TO 947
110 READ J
120 POKE I , J
130 NEXT
140 PRINT"SYS 880 WILL ENABLE AND DISABLE
150 PRINT"        THE AUTO REPEAT FUNCTION
160 END
```

Block Get Routine

Bill MacLean
BMB Compuscience, Canada

The machine language routine below can be used with direct access routines to transfer the contents of a disk buffer into PET memory.

```
100 FOR J = 826 TO 914
110 READ A
120 POKE J , A
130 NEXT
200 DATA 169 ,   0 , 133 ,  52 , 169 , 127
210 DATA 133 ,  53 ,  32 , 248 , 205 ,  32
220 DATA 159 , 204 ,  32 , 210 , 214 , 165
230 DATA  18 , 240 ,   3 ,  76 ,   3 , 206
240 DATA 165 ,  17 , 133 , 210 , 169 ,   0
250 DATA 133 ,   1 , 169 , 127 , 133 ,   2
260 DATA 166 , 210 ,  32 , 198 , 255 ,  32
270 DATA 207 , 255 , 201 ,  10 , 240 , 249
280 DATA 201 ,  13 , 240 ,   8 , 160 ,   0
290 DATA 145 ,   1 , 230 ,   1 , 208 , 237
300 DATA 165 , 210 ,  32 , 204 , 255 ,  32
310 DATA 248 , 205 ,  32 , 159 , 204 , 160
320 DATA   0 , 165 ,   1 , 145 ,  68 , 200
330 DATA 169 ,   0 , 145 ,  68 , 200 , 169
340 DATA 127 , 145 ,  68 ,  96 ,  66
```

Note: For 16K machines, change the three 127's to 63's.

The command to use this program is:

```
SYS 826 , LF# , A$
```

It replaces:

```
INPUT# (LF#), A$
```

It works with ASCII string files only. Any string variable can be used but must be initialized before calling.

```
eg. A$ = "" : SYS 826 , 2 , A$
```

A$ only has to be initialized once.

Since the string is transfered from disk into a dummy input buffer ($7F00 to $8000 on 32K machines, $3F00 to 4000 on 16K), it is necessary to move the record into BASIC string storage. This can be accomplished by:

```
A$ = A$ + ""    or    Y$ = A$ + ""    or    A$(J) = A$ + ""
```

This routine has two advantages over INPUT#. It permits inputting strings up to 255 characters and it strips the Line Feeds placed on disk by PRINT#. Also it is much faster than using GET# in a loop.

Corrections to Transactor #6

In Jim Butterfield's Disk Peek program the print was smudged out in lines 430, 440 and 450. The lines read:

```
430 Y(0)=0:Y(4)=Y(4)+8:J=4
440 IFY(J)>15THENY(J)=Y(J)-16:J=J-1:....
450 PRINT:PRINT"  ";:FORJ=1TO4:....
```

Also, for some strange reason, line 340 may have to be modified for correct operation depending on your machine. Move the second statement of line 340 (GET#1....) to line 345 and change 340 to:

```
340 PRINT#1,"M-R";CHR$(4);CHR$(16);CHR$(1);CHR$(224)
```

Line 235 in Mike Casey's "Tape Index" program should read after the second semi-colon:

```
....INT((L(N,5)-L(N,1))/60);"SEC....
```

kobetek
systems
limited
RR #1 WOLFVILLE NOVA SCOTIA
CANADA B0P 1X0 (902)542-9100

December 1, 1979

DIAL-A-ROM™

Provides manual switching between six 2716 ROMs (Toolkit, Word pro II, etc...) or six 2708's.

Comes in attractive metal housing, with 24 pin plug on flat ribbon cable, to fit into one of the empty ROM sockets in 16/32K PETs.

Delivered as 2716 switcher, with simple instructions and jumpers to change to 2708.

Model 6: $99.00
Model 6a: $120.00
with 25-pin connector for quick disconnect when moving the PET.
Model 18 (Jan-Feb '80)
Combined software select and manual switch for 18 ROMs.

DIAL-A-ROM™ is a product of KOBETEK SYSTEMS LTD. of Wolfville.

Although it was designed with the Commodore PET in mind, it is of use in any application requiring the switching of 2716's or 2708's.

*** Watch for more products from KOBETEK ***

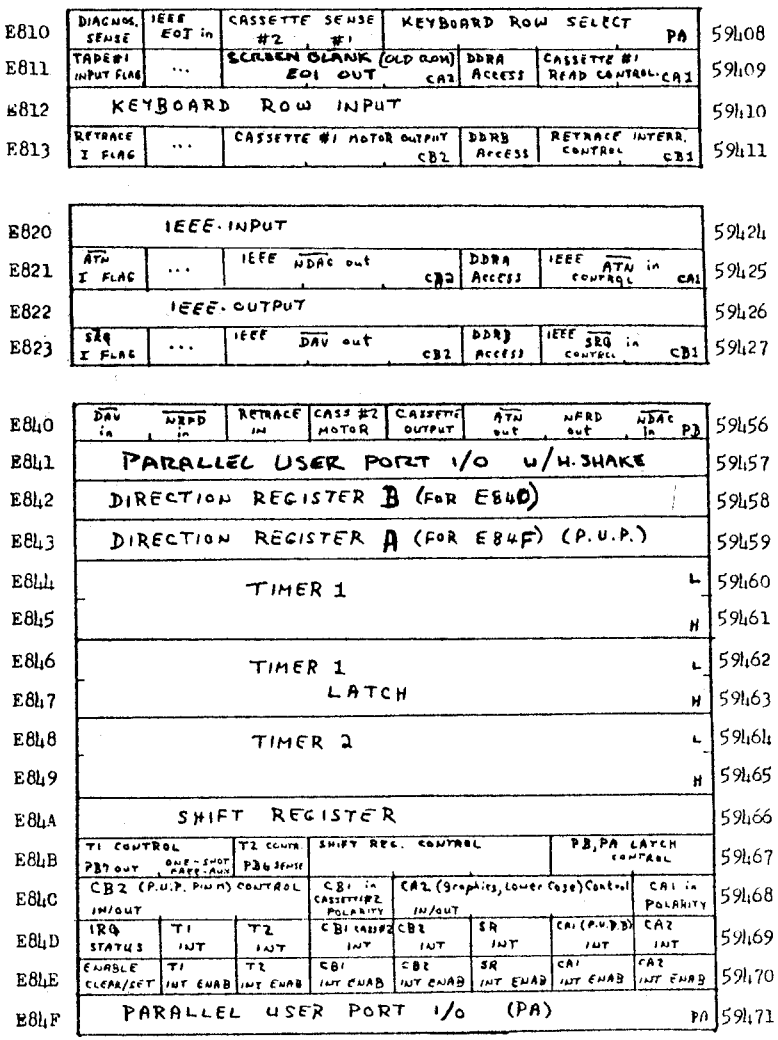
Coming soon : PET interface for the Terrapin Turtle™
two-way RS232 interface for the PET

All orders: Add $ 3.00 for shipping.

0000-0002	0-2	USR Jump instruction
0003	3	Current I/O Device for prompt-suporess
0005	5	Cursor position for Input & Print
0008-0009	8-9	Integer address from Basic (for SYS, GOTO, etc.)
000A-0059	10-89	Basic input buffer; # of array subscripts
005A	90	Search character (usually ':' or end-of-line)
005B	91	Scan-between-quotes flag
005C	92	Basic input buffer pointer; number of subscripts
005D	93	First-character of array-name; default DIM flag
005E	94	Type: FF=string; 00=numeric
005F	95	Type: 80=integer; 00=floating point
0060	96	'DATA' scan flag; LIST quote flag; memory flag
0061	97	Subscript flag; FNx flag
0062	98	0=input, 64=get, 152=read (flag)
0063	99	flag for trigonometric signs/comparison evaluation flag
0064	100	input flag (suporess output if negative)
0065	101	variable pseudo-stack pointer
0066	102	fixed-point pseudo-stack pointer
0067	103	dummy value (0)
0068-0070	104-112	variable x pseudo-stack
0071-0072	113-114	pointer for number transfer
0073-0074	115-116	number pointer
0075-0078	117-120	product staging area for multiplication
007A-007B	122-123	start of basic pointer
007C-007D	124-125	end of basic/start of varibles pointer
007E-007F	126-127	end of variables/startof arrays
0080-0081	128-129	start of available space pointer
0082-0083	130-131	bottom of strings (moving down) pointer
0084-0085	132-133	top of strings (moving down) pointer
0086-0087	134-135	limit of Basic memory pointer
0088-0089	136-137	current program line number
008A-008B	138-139	previous line number
008C-008D	140-141	previous line address (for CONT)
008E-008F	142-143	line number of DATA line
0090-0091	144-145	memory address of DATA line
0092-0093	146-147	input vector (DATA etc.)
0094-0095	148-149	current variable name
0096-0097	150-151	current variable address
0098-0099	152-153	variable pointer for current FOR/NEXT
009A	154	Y save register; new operator save
009C	156	comparison symbol accumulator: <1 =2 >4
009D-00A1	157-161	number work area for SQR, etc.
00A2	162	pseudo-stack yardstick (3 or 7)
00A3-00A5	163-165	jump vector for functions
00A6-00AA	166-170	numeric store area
00AB-00AF	171-175	numeric store area
00B0-00B5	176-181	primary accumulator E,M,M,M,M,S
00B6	182	Taylor series constant counter
00B7	183	accumulator high-order propogation word
00B8-00BD	184-189	secondary accumulator
00BE	190	sign comparison, primary/secondary
00BF	191	low-order rounding byte for primary acc

00C0-00C1	192-193	Cassette buffer length/Taylor constant pointer
00C2-00D9	194-217	Subrtn: Get Basic Char; C9,CA=pointer
00DA-00DE	218-222	RND storage and work area
00E0-00E1	224-225	Pointer to screen cursor line
00E2	226	Poisition of cursor on line
00E3-00E4	227-228	Utility pointer; tape buffer,scrolling
00E5-00E6	229-230	End of current program/tape end address
00E7-00E8	231-232	Tape timing constants
00E9	233	Tape buffer character
00EA	234	Direct/programmed cursor; 00=direct
00EB	235	Tape read/verify flag
00EC	236	Tape write flag
00ED	237	
00EE	238	Number of k characters in file name
00EF	239	Logical file number
00F0	240	File command (from OPEN)
00F1	241	Device number
00F2	242	Maximum line length (40 or 80)
00F3-00F4	243-244	Tape buffr address (start of buffer)
00F5	245	Line where cursor lives
00F6	246	Last key pushed (ASCII); buffer checksum
00F7-00F8	247-248	Tape start address/tape pointer
00F9-00FA	249-250	File name pointer
00FB	251	Number of "insert" keys pushed
00FC	252	Serial bit shift word
00FD	253	# blocks remaining to write
00FE	254	Serial word buffer
0100-010A	256-266	Binary to ASCII conversion area
010B-01FF	267-511	Stack area
0200-0202	512-514	TI and TI$ clock - jiffies
0203	515	Which key depressed: 255 = no key
0204	516	Shift key: 1 if depressed
0205-0206	517-518	Clock (unused?)
0207	519	Cassette 1 status switch
0208	520	Cassette 2 status switch
0209	521	Keyswitch PIA: STOP & RVS flags, etc.
020A	522	
020B	523	Load=0, Verify=1
020C	524	Status
020D	525	# characters in keyboard buffer
020E	526	Reverse flag
020F-0218	527-536	Keyboard buffer
0219-021A	537-538	Hardware interrupt vector
021B-021C	539-540	Break interrupt vector
021D	541	
021E	542	End-of-line-for -input pointer
0220-0221	544-545	Cursor log (row, column)
0222	546	PBD image for tape I/O
0223	547	Key image
0224	548	0=flashing cursor; else no cursor shows
0225	549	Cursor timing countdown
0226	550	Character under cursor
0227	551	Cursor blink flag
0228	552	Tape write
0229-0241	553-577	Line address high & screen line wrap table

02h2-02hB	578-587	Logical numbers of open files
02hC-0255	588-597	Device numbers of open files
0256-025F	598-607	Command/Secondary address of open files
0260	608	Input from screen/input from keyboard
0261	609	X-save flag
0262	610	How many open files
0263	611	Input device, normally 0
0264	612	Output CMD device, normally 3
0265	613	Tape parity
0266	614	
0268	616	Pointer in filename transfer
026A	618	
026C	620	Serial bit count
026F	623	
0270	624	Tape write countdown
0271	625	Tape buffer #1 count
0272	626	Tape buffer #2 count
0273	627	Leader counter
0274	628	Flag for tape error
0275	629	0 if lst ½-byte cntr not written
0276	630	2nd ½-byte cntr/tape error count
0277	631	
0278	632	Cassette read flag
0279	633	Checksum working word
027A-0339	634-825	Tape #1 buffer
033A-03F9	826-1017	Tape #2 buffer
0400-7FFF	1024-32767	Available RAM including expansion
8000-8FFF	32768-36863	Video RAM
9000-BFFF	36864-49151	Available ROM expansion area
C000-E077	49152-57463	Microsoft Basic
E078-E7F8	57464-59384	Keyboard/Screen/Interrupt monitor
E810	59408	PIA1 - Keyboard A register; (Direction with CRA2=1)
E811	59409	PIA1 - Keyboard A control
E812	59410	PIA1 - Keyboard B register; (Direction with CRB2=1)
E813	59411	PIA1 - Keyboard B control
E820	59424	PIA2 - IEEE A register; (Direction with CRA2=1)
E821	59425	PIA2 - IEEE A control
E822	59426	PIA2 - IEEE B register; (Direction with CRB2=1)
E823	59427	PIA2 - IEEE B control
E840	59456	VIA I/O register B
E841	59457	VIA I/O register A with handshake
E842-E843	59458-59459	VIA Data Direction regs, A and B
E844-E845	59460-59461	VIA Timer 1
E846-E847	59462-59463	VIA Timer 1 latch
E848-E849	59464-59465	VIA Timer 2
E84A	59466	VIA shift register
E84B	59467	ACS: T1.T1.T2.SR.SR.SR.PB.PA
E84C	59468	PCR: B2.B2.B2.B1.A2.A2.A2.A1
E84D-E84E	59469-59470	IFR, IER: T1.T2.CB1.BC2.SR.CA1.CA2
E84F	59471	I/O Register A without handshake
F000-FFFF	61440-65535	Reset/tape/diagnostic monitor



A few routines from PET BASIC — Compiled by Jim Butterfield, Toronto

(Original ROM)

C2AC-C2D9 peeks at the stack for an active FOR loop
C2DA-C31C 'opens up' a space in Basic for insertion of a new linex.
C31D-C329 tests for stack-too-deep and aborts if found.
C32A-C356 check available memory space
C357-C388 sends a canned error message from C190 area, then drops into:
C389-C391 Signals 'ready'
C394-C3A9 gets a line of input, analyzes it, executes it
C3AC-C42E handles a new line of Basic from keyboard: deletes old line, etc.
C430-C460 corrects the chaining between Basic lines after insert/delete
C462-C476 receives a line from the keyboard into the Basic buffer
C479-C48C gets each character from keyboard
C48D-C521 looks up the keywords in an input lines and changes to "tokens"
C522-C550 searches for the location of a Basic line from number in 8,9
C551-C599 implements NEW command - clears everything (011/019 ROM change)
C59A-C5A7 sets the Basic pointer to start-of-program
C5A8-C647 performs LIST command
C649- C68F executes a FOR statement
C692-C6B4 continues to build FOR vectors
C6B5-C6EF reads and executes the next Basic statement, finds next line, etc.
C6F2-C70A executes the Basic Command as a subroutine
C70D-C71B performs RESTORE
C71C-C742 handles STOP, END, and BREAK procedures.
C745-C75E performs CONT
C75F-C76D set pause after carriage return (never called)
C770-C772 performs CLR
C775-C77D performs RUN
C780-C79A performs GOSUB
C79D-C7C9 performs GOTO
C7CA-C7FD performs RETURN
C7FE-C81E scans for start of next Basic Line
C820-C840 performs IF
C843-C862 performs ON
C863-C89A gets a fixed-point number from Basic and stores in 8,9
C89D-C91B performs LET
C91C-C97E check numeric digit/move string pointer
C97F-C982 performs PRINT#
C985-C996 performs CMD
C999-CA24 performs PRINT
CA27-CA41 prints string from address in Y, A
CA44-CA76 prints a character
CA77-CA9E handles bad input data
CA9F-CAC5 performs GET
CAC6-CADF performs INPUT#
CAE0-CB14 performs INPUT
CB17-CB21 prompts and receives the input
CB24-CC11 performs READ
CC12-CC35 canned messages: EXTRA IGNORED; REDO FROM START
CC36-CC8F performs NEXT
CC92-CCB5 checks Basic format, data type, flags TYPE MISMATCH
CCB8-CD38 inputs and evaluates any expression (numeric or string)
CD3a-CD9C pushes a partially-evaluated argument to the stack
CD9D-CDB9 evalues a numeric, variable, or pi, or identifies other symbol
CDBC-CDC0 value of pi in floating binary
CDC1-CDE7 checks for special characters (+,-,",.) at start of expression
CDE8-CDF6 performs NOT function
CDF7-CE04 checks for various functions
CE05 evaluates expression within parentheses ()
CE08 checks for right parenthesis)
CE0E checks for left parenthesis (
CE11-CE1B checks for comma
CE1C-CE20 prints SYNTAX ERROR and exits
CE21-CE27 sets up function for future evaluation
CE28-CE39 set up a variable name search
CE3B-CE96 checks for special variables TI, TI$, and ST
CE97-CED5 identifies and sets up function references
CED6-CF05 perform the OR and AND functions
CF06-CF6D performs comparisons
CF6E-CF7A sets up DIM execution
CF7B-D00E searches for a Basic variable
D00F-D078 creates a new Basic variable
D079-D087 logs Basic variable location
D088-D098 is array pointer subroutine
D099-D09C is 32768 in floating binary
D09D-D0B8 is floating point-to-fixed conversion for signed values
D0B9-D263 locates and/or creates arrays
D264-D277 performs FRE function
D278-D284 converts fixed point-to-floating
D285-D28A performs POS function
D28B-D294 checks direct/indirect command, gives 'ILLEGAL DIRECT'
D295-D348 executes DEF statements and evaluation FNx
D349-D36A performs STR$ function
D36B-D3D1 scans and sets up string elements
D3D2-D403 builds string vectors
D404-D5C3 does 'garbage collection' - discards unwanted strings
D5C4-D5D7 performs CHR$ function
D5D8-D653 performs LEFT$, RIGHT$, MID$ functions
D654-D662 performs LEN, gets string length
D663-D672 performs ASC function
D673-D684 gets a single-byte value from Basic
D685-D6C3 evaluates VAL function
D6C4-D6CF gets two arguments (16-bit and 8-bit) from Basic
D6D0-D6E5 checks argument is in range 0-65535
D6E6-D701 performs PEEK and POKE
D702-D71D executes WAIT statement
D71E-D890 performs addition and subtraction
D891-D8BE contains floating-point constants
D8BF-D8FC performs LOG function
D8FD-D95D performs multiplication
D95E-D988 loads secondary accumulator from memory ($B8 to $BD)
D989-D9B3 test and adjust primary/secondary accumulators
D9B4-D9E0 routines to multiply or divide by 10
D9E1-DA73 performs division
DA74-DA98 loads primary accumulator from memory ($b0-$B5)
DA99-DACD transfers primary accumulator to memory
DACE-DADD transfers secondary accumulator to primary
DADE-DAEC transfers primary accumulator to secondary
DAED-DAFC rounds the primary accumulator
DAFD-DB29 extracts primary sign; performs SGN function
DB2A-DB2C performs ABS
DB2D-DB6C compares primary accumulator to memory

DB6D-DB9D Convert Floating point to fixed, unsigned
DB9E-DBC4 perform INT function
DBC5-DC4F convert ASCII string to floating point
DC50-DC84 get new ASCII digit
DC94-DCAE print Basic Line number
DCAF-DDE2 convert floating point to ASCII string (at 0100 up)
DDE3-DE23 conversion constants - decimal or clock
DE24-DE2D evaluation SGR function
DE2E-DE66 evaluation of power function
DE67-DE71 negate (monadic -)
DEA0-DEF2 perform EXP function
DEF3-DF3C perform function series evaluation
DF45-DF9D perform RND calculation
DF9E evaluate COS function
DFA5-DFED evaluate SIN function
DFEE-E019 evaluate TAN function
E048-E077 evaluate ATN function
E0B5-E0CC Basic scan program, transferred to 00C2-00D9
E0D2-E173 completion of power-on-reset; memory test, etc.
E19B-E1BB partial test for TI and TI$
E1BC-E1E0 input/read/get director
E1E1-E27C initialize I/O registers, clear screen, reset subroutine
E27D-E3C3 receive input from keyboard/screen
E3C4-E3E9 set up new screen line
E3EA-E52F output character to screen
E530-E5DA check for and perform screen scrolling
E5DB-E66A start new screen line
E66B-E67D interrupt entry
E67E-E683 interrupt return
E685-E73E hardware interrupt routine: cursor flash, tape motor, keyboard
E73F-E7AB convert keyboard matrix to ASCII
E7AC-E7B9 write-on-screen subroutine
E7DE-E7EB print canned monitor message
F0B6-F1CB IEEE-488 channel open, test, close
F1CC-F22F get input character from keybaord, screen cassette, IEEE
F230-F27C output character to screen, cassette, IEEE
F27D-F2A3 restore normal I/O, clear IEEE channels
F2A4-F2AA abort (not close!) all files
F2AB-F2B7 locate logical file table entry
F2B8-F2C7 transfer file table entries to Device, Command
F2C8-F329 perform file CLOSE
F32A-F33E test stop key
F33F-F345 test if direct/indirect command for suppressing file advice
F346-F3FE perform file LOAD
F3FF-F421 print "SEARCHING .. "
F422-F432 print "LOADING .. " or "VERIFYING"
F433-F461 get parameters for LOAD and SAVE
F462-F494 perform IEEE sequences for LOAD, SAVE, and OPEN
F495-F4BA search for specific tape header
F4BB-F4D3 perform VERIFY
F4D4-F529 get parameters for OPEN and CLOSE
F52A-F5AD perform OPEN
F5AE-F5E2 search for any tape header
F5E3-F5EC clear tape buffer
F5ED-F64C write tape header
F64D-F666 get start & end addresses from tape header
F667-F67C Set buffer start address
F67D-F694 set tape buffer start and end pointers
F695-F69D perform SYS command
F69E-F71E perform SAVE
F71C-F735 find unused secondary address
F736-F78A update clock
F78B-F7DB set input device
F7DC-F82C set output device
F82D-F83A bump tape buffer counter
F83B-F85D wait for cassette PLAY switch
F85E-F870 test cassette switch line
F871-F87E wait for cassette RECORD and PLAY switches
F87F-F8B8 read tape initiation routine
F8B9-F8D1 write tape initiation routine
F8D2-F912 complete tape read or write
F913-F91D wait for I/O completion
F91E-F92D test stop key and abort if necessary
F92E-F95E subroutine to set tape read timing
F95F-FBDB interrupt routine for tape read
FBDC-FBE4 save memory pointer
FBE5-FBEB set ST error flag
FBEC-FBFF subroutine to count 8 serial bits per byte
FC00-FC1B subroutine to write a bit to tape
FC1C-FCFA interrupt 1 for tape write - entry at FC21
FCFB-FD15 terminate I/O and restore normal vectors
FD16-FD37 subroutine to set interrupt vector
FD38-FD47 power-on reset entry; test for diagnostic
FD48-FD7B diagnostic routine
FD7C-FD8F checksum routine
FD90-FD9A pointer advance subroutine
FD9B-FFB1 diagnostic routines

JUMP TABLE:

FFC0	OPEN
FFC3	CLOSE
FFC6	set input device
FFC9	set output device
FFCC	restore normal I/O devices
FFCF	input character (from screen)
FFD2	output character
FFD5	LOAD
FFD8	SAVE
FFDB	VERIFY
FFDE	SYS
FFE1	test stop key
FFE4	get character from keyboard buffer
FFE7	abort all I/O channels
FFEA	update clock

FFED-FFFA turn off cassette motors
FFFA-FFFB NMI vector (mangled)
FFFC-FFFD reset vector
FFFE-FFFF interrupt vector

Memory locations for ROM upgrade on PET computers

Jim Butterfield, Toronto

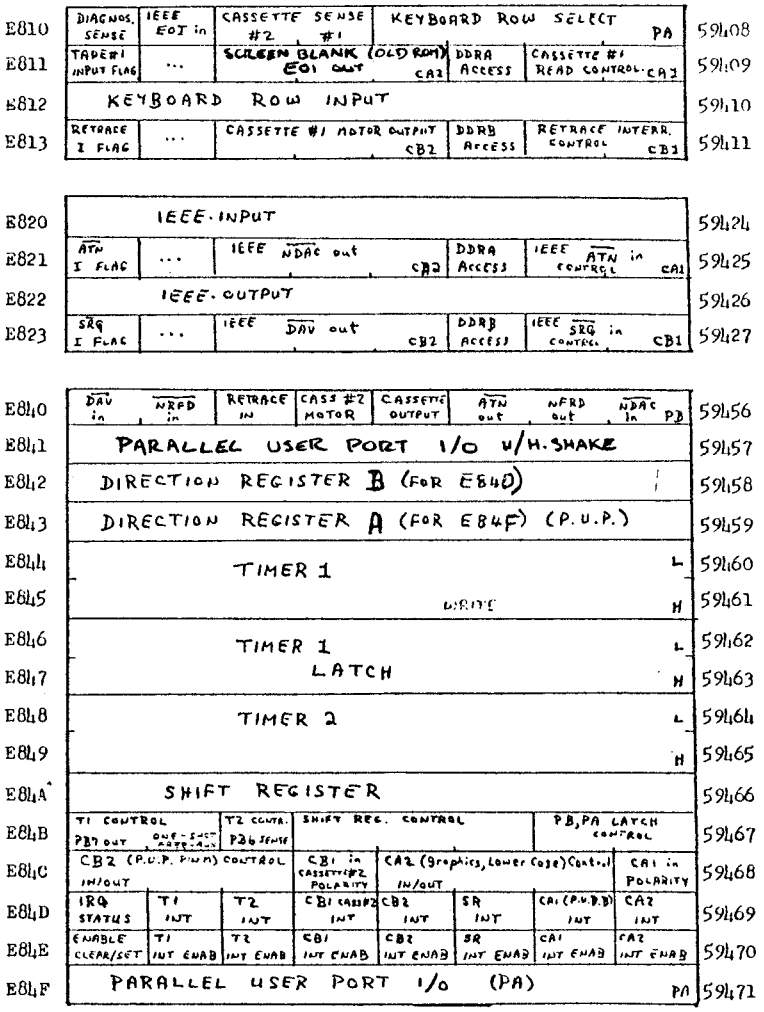
0000-0002	0-2	USR Jump instruction
0003	3	Search character
0004	4	Scan-between-quotes flag
0005	5	Basic input buffer pointer; # subscripts
0006	6	Default DIM flag
0007	7	Type: FF=string, 00=numeric
0008	8	Type: 80=integer, 00=floating point
0009	9	DATA scan flag; LIST quote flag; memory flag
000A	10	Subscript flag; FNx flag
000B	11	0=input; 64=get; 152=read
000C	12	ATN sign flag; comparison evaluation flag
000D	13	input flag; suppress output if negative
000E	14	current I/O device for prompt-suppress
0011-0012	17-18	Basic integer address (for SYS, GOTO etc)
0013	19	Temporary string descriptor stack pointer
0014-0015	20-21	Last temporary string vector
0016-001E	22-30	Stack of descriptors for temporary strings
001F-0020	31-32	Pointer for number transfer
0021-0022	33-34	Misc. number pointer
0023-0027	35-39	Product staging area for multiplication
0028-0029	40-41	Pointer: Start-of-Basic memory
002A-002B	42-43	Pointer: End-of-Basic, Start-of-Variables
002C-002D	44-45	Pointer: End-of-Variables, Start-of-Arrays
002E-002F	46-47	Pointer: End-of-Arrays
0030-0031	48-49	Pointer: Bottom-of-Strings (moving down)
0032-0033	50-51	Utility string pointer
0034-0035	52-53	Pointer: Limit of Basic Memory
0036-0037	54-55	Current Basic line number
0038-0039	56-57	Previous Basic line number
003A-003B	58-59	Pointer to Basic statement (for CONT)
003C-003D	60-61	Line number, current DATA line
003E-003F	62-63	Pointer to current DATA item
0040-0041	64-65	Input vector
0042-0043	66-67	Current variable name
0044-0045	68-69	Current variable address
0046-0047	70-71	Variable pointer for FOR/NEXT
0048	72	Y save register; new-operator save
004A	74	Comparison symbol accumulator
004B-004C	75-76	Misc numeric work area
004D-0050	77-80	Work area; garbage yardstick
0051-0053	81-83	Jump vector for functions
0054-0058	84-88	Misc numeric storage area
0059-005D	89-93	Misc numeric storage area
005E-0063	94-99	Accumulator#1: E,M,M,M,M,S
0064	100	Series evaluation constant pointer
0065	101	Accumulator hi-order propogation word
0066-006B	102-107	Accumulator#2
006C	108	Sign comparison, primary vs. secondary
006D	109	low-order rounding byte for Acc#1
006E-006F	110-111	Cassette buffer length/Series pointer

Memory map, upgrade ROM, contd.

0070-0087	112-135	Subrtn: Get Basic Char; 77,78=pointer
0088-008C	136-140	RND storage and work area
008D-008F	141-143	Jiffy clock for TI and TI$
0090-0091	144-145	Hardware interrupt vector
0092-0093	146-147	Break interrupt vector
0094-0095	148-149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key depressed: 255=no key
0098	152	Shift key: 1 if depressed
0099-009A	153-154	Correction clock
009B	155	Keyswitch PIA: STOP and RVS flags
009C	156	Timing constant buffer
009D	157	Load=0, Verify=1
009E	158	# characters in keyboard buffer
009F	159	Screen reverse flag
00A0	160	IEEE-488 output flag: FF=character waiting
00A1	161	End-of-line-for-input pointer
00A3-00A4	163-164	Cursor log (row, column)
00A5	165	IEEE-488 output character buffer
00A6	166	Key image
00A7	167	0=flashing cursor, else no cursor
00A8	168	Countdown for cursor timing
00A9	169	Character under cursor
00AA	170	Cursor blink flag
00AB	171	EOT bit received
00AC	172	Input from screen/input from keyboard
00AD	173	X save flag
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00B1	177	Tape character parity
00B2	178	Byte received flag
00B4	180	Tape buffer character
00B5	181	Pointer in filename transfer
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Countdown for tape write
00BB	187	Tape buffer#1 count
00BC	188	Tape buffer#2 count
00BD	189	Write leader count; Read pass1/pass2
00BE	190	Write new byte; Read error flag
00BF	191	Write start bit; Read bit seq error
00C0	192	Pass 1 error log pointer
00C1	193	Pass 2 error correction pointer
00C2	194	0=Scan; 1-15=Count; $40=Load; $80=End
00C3	195	Checksum for read; Leader length for write
00C4-00C5	196-197	Pointer to screen line
00C6	198	Position of cursor on above line

Memory map, upgrade ROM, contd.

00C7-00C8	199-200	Utility pointer: tape buffer, scrolling
00C9-00CA	201-202	Tape end address/end of current program
00CB-00CC	203-204	Tape timing constants
00CD	205	00=direct cursor, else programmed cursor
00CE	206	Timer 1 enabled for tape read; 00=disabled
00CF	207	EOT signal received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current logical file number
00D3	211	Current secondary addrs, or R/W command
00D4	212	Current device number
00D5	213	Line length (40 or 80) for screen
00D6-00D7	214-215	Start of tape buffer, address
00D8	216	Line where cursor lives
00D9	217	Last key input; buffer checksum; bit buffer
00DA-00DB	218-219	File name pointer
00DC	220	Number of keyboard INSERTs outstanding
00DD	221	Write shift word/Receive input character
00DE	222	#blocks remaining to write/read
00DF	223	Serial word buffer
00E0-00F8	224-248	Screen line table: hi order address & line wrap
00F9	249	Cassette#1 status switch
00FA	250	Cassette#2 status switch
00FB-00FC	251-252	Tape start address
0100-010A	256-266	Binary to ASCII conversion area
0100-013E	256-318	Tape read error log for correction
0100-01FF	256-511	Processor stack area
0200-0250	512-592	Basic input buffer
0251-025A	593-602	Logical file number table
025B-0264	603-612	Device number table
0265-026E	613-622	Secondary address, or R/W cmd, table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 buffer
033A-03F9	826-1017	Tape#2 buffer
03FA-03FB	1018-1019	Vector for Machine Language Monitor
0400-7FFF	1024-32767	Available RAM including expansion
8000-8FFF	32768-36863	Video RAM
9000-BFFF	36864-49151	Available ROM expansion area
C000-E0F8	49152-57592	Microsoft Basic interpreter
E0F9-E7FF	57593-59391	Keyboard, Screen, Interrupt programs
E810-E813	59408-59411	PIA1 - Keyboard I/O
E820-E823	59424-59427	PIA2 - IEEE488 I/O
E840-E84F	59456-59471	VIA - I/O and Timers
F000-FFFF	61440-65535	Reset, tape, diagnostic monitor



Routines in Upgrade ROM Jim Butterfield, Toronto

C000-C045 Action addresses for primary keywords
C046-C073 Action addresses for functions
C074-C091 Hierarchy and action addresses for operators
C092-C192 Table of Basic keywords
C193-C2A9 Basic messages, mostly error messages.
C2AA-C2D7 Search stack for FOR or GOSUB activity
C2D8-C31A Open up space in memory
C31B-C327 Test: stack too deep?
C328-C354 Check available memory
C355 Send canned error message, then:
C389-C3AA Print READY.
C3AB-C441 Handle new Basic line from keyboard
C442-C46E Rebuild chaining of Basic lines in memory.
C46F-C494 Receive line from keyboard
C495-C52B Change keywords to Basic tokens
C52C-C55A Search Basic for a given Basic line number
C55B Perform NEW, then:
C577-C5A6 Perform CLR
C5A7-C5B4 Reset Basic execution to start-of-program
C5B5-C657 Perform LIST
C658-C6FF Perform FOR
C700-C72F Execute Basic statement
C730-C73E Perform RESTORE
C73F-C76A Perform STOP and END
C76B-C784 Perform CONT
C785-C78F Perform RUN
C790-C7AC Perform GOSUB
C7AD-C7D9 Perform GOTO
C7DA Perform RETURN, and perhaps:
C7F3-C80D Perform DATA, i.e., skip rest of statement
C80E Scan for next Basic statement
C811-C82F Scan for next Basic line
C830 Perform IF, and perhaps:
C843-C852 Perform REM, i.e., skip rest of line
C853-C872 Perform ON
C873-C8AC Get fixed-point number from Basic
C8AD-C927 Perform LET
C928-C936 Add ASCII digit to accumulator #1
C937-C98A Continue to perform LET
C98B-C990 Perform PRINT#
C991-C9A4 Perform CMD
C9A5-CA1B Perform PRINT
CA1C-CA38 Print string from memory
CA39-CA4E Print single format character (space, cursor-right, ?)
CA4F-CA7C Handle bad input data
CA7D-CAA6 Perform GET
CAA7-CAC0 Perform INPUT#
CAC1-CAF9 Perform INPUT
CAFA-CB06 Prompt and receive input
CB07-CBFB Perform READ; common routines used by INPUT and GET
CBFC-CC1F Messages: EXTRA IGNORED, REDO FROM START
CC20-CC78 Perform NEXT
CC79-CC9E Checks data type, prints TYPE MISMATCH
CC9F Inputs & evaluates any expression (numeric or string)

CDEC Evaluate expression within parentheses ()
CDF2 Check right parenthesis)
CDF5 Check left parenthesis (
CDF8-CE02 Check for comma
CE03-CE07 Print SYNTAX ERROR and exit
CE08-CE0E Set up function for future evaluation
CE0F-CE88 Search for variable name
CE89-CEC7 Identify and set up function references
CEC8 Perform OR
CECB-CEF7 Perform AND
CEF8-CF5F Perform comparisons, string or numeric
CF60-CF6C Perform DIM
CF6D-CFF6 Search for variable location in memory
CFF7-D000 Check if ASCII character is alphabetic
D001-D077 Create new Basic variable
D078-D088 Array pointer subroutine
D089-D08C 32768 in floating binary
D08D-D0AB Evaluate expression for positive integer
D0AC-D227 Find or create array
D228-D258 Compute array subscript size
D259 Perform FRE
D26D-D279 Converts fixed-point to floating-point
D27A-D27F Perform POS
D280-D28C Check if direct command, print ILLEGAL DIRECT
D28D-D2BA Perform DEF
D2BB-D2CD Check FNx syntax
D2CE-D33E Evaluate FNx
D33F-D34E Perform STR$
D34F-D360 Calculate string vector
D361-D3CD Scan and set up string
D3CE-D3FF Subroutine to build string vector
D400-D496 Garbage collection subroutine
D497-D4DF Check for most eligible string for collection
D4E0-D516 Collect a string
D517-D553 Perform string concatenation
D554-D57C Build string into memory
D57D-D5B4 Discard unwanted string
D5B5-D5C5 Clean the descriptor stack
D5C6-D5D9 Perform CHR$
D5DA-D605 Perform LEFT$
D606-D610 Perform RIGHT$
D611-D63A Perform MID$
D63B-D655 Pull string function parameters from stack
D656-D65B Perform LEN
D65C-D664 Move from string-mode to numeric-mode
D665-D674 Perform ASC
D675-D686 Input byte parameter
D687-D6C5 Perform VAL
D6C6-D6D1 Get two parameters for POKE or WAIT
D6D2-D6E7 Convert floating-point to fixed-point
D6E8-D706 Perform PEEK
D707-D70F Perform POKE
D710-D72B Perform WAIT
D72C-D732 Add 0.5 to accumulator#1
D733-D744 Perform subtraction
D745-D76D Microsoft joke

D76E-D852 Perform addition
D853-D889 Complement accumulator#1
D88A-D88E Print OVERFLOW and exit
D88F-D8C7 Multiply-a-byte subroutine
D8C8-D8F5 Function constants: 1, SQR(.5), SQR(2), -0.5, etc.
D8F6 Perform LOG
D937-D964 Perform multiplication
D965-D997 Multiply-a-bit subroutine
D998-D9C2 Load accumulator #2 from memory
D9C3-D9DF Test and adjust accumulators #1 and #2
D9E0-D9ED Handle overflow and underflow
D9EE-DA04 Multiply by 10
DA05-DA09 10 in floating binary
DA0A Divide by 10
DA13 Perform divide-into
DA1E-DAAD Perform divide-by
DAAE-DAD2 Load Accumulator #1 from memory
DAD3-DB07 Store accumulator #1 into memory
DB08-DB17 Copy accumulator #2 into accumulator #1
DB18-DB26 Copy accumulator #1 into accumulator #2
DB27-DB36 Round off accumulator #1
DB37-DB44 Compute SGN value of accumulator #1
DB45-DB63 Perform SGN
DB64-DB66 Perform ABS
DB67-DBA6 Compare accumulator #1 to memory
DBA7-DBD7 Convert floating-point to fixed-point
DBD8-DBFE Perform INT
DBFF-DC89 Convert string to floating-point
DC8A-DCBE Get new ASCII digit
DCBF-DCCD String conversion constants: 99999999, 999999999, 1E+9
DCCE Print IN, followed by:
DCD9-DCE8 Print Basic line number
DCE9-DE1C Convert number or TI$ to ASCII
DE1D-DE5D Constants for numeric conversion
DE5E Perform SQR
DE68 Perform power function
DEA1-DEAB Perform negation
DEAC-DED9 Constants for string evaluation
DEDA-DF2C Perform EXP
DF2D-DF76 Function series evaluation subroutines
DF77-DF7E Manipulation constants for RND
DF7F-DFD7 Perform RND
DFD8 Perform COS
DFDF-E027 Perform SIN
E028-E053 Perform TAN
E054-E08B Constants for trig evaluation: pi/2, 2*pi, .25, etc.
E08C-E0BB Perform ATN
E0BC-E0F8 Constants for ATN series evaluation
E0F9-E110 Subroutine to be moved to zero page ($70 to $87)
E111-E115 Initial RND seed
E116-E1B6 Initialize Basic system
E1B7-E1DD Messages: BYTES FREE, ### COMMODORE BASIC ###
E1DE Initialize I/O registers, and:
E229 Clear screen, and:
E257-E284 Home cursor
E285-E2F3 Input from screen or keyboard; wait for input completion
E2F4-E33E Input from screen
E33F-E34B Test for quotation mark and reverse quote-flag
E34C-E38A Set up screen print parameters
E38B-E395 Prevent 80-character line from getting any longer
E396-E3B3 Extend 40-character line to 80 characters
E3B4-E3D7 Back into the previous line (via DEL or CURSOR LEFT key)
E3D8-E518 Handle ASCII character for screen output
E519-E53E Go to next line on screen
E53F-E5B9 Scroll the screen
E5BA-E61A Open a line on the screen (via INSERT key)
E61B-E62D Main interrupt entry point
E62E-E6E9 Hardware interrupt: service clock, keyboard, cassettes
E6EA-E6F7 Print character on screen
E6F8-E769 Table: decoder for keyboard matrix
E76A-E796 MLM subroutine: output hex digits
E797-E7A6 MLM subroutine: swap TMP0 and TMP2
E7A7-E7F6 MLM subroutine: input hex digits
E7F7-E7FF MLM subroutine: print ?
F000-F0B5 Monitor messages, mostly for Input/Output
F0B6 Set up IEEE for Listen, Talk, etc.
F0EE-F127 Send character to IEEE-488 bus
F128-F135 Output character immediate mode to IEEE-488
F136-F155 Send errors: WRITE TIMEOUT, DEVICE NOT PRESENT, etc.
F156-F163 Send canned I/O message
F164-F16E Send immediate Listen command, then secondary address
F16F-F17E Output character deferred mode to IEEE-488
F17F-F18B Drop IEEE channel: send Unlisten or Untalk
F18C-F1D0 Input character from IEEE-488 bus
F1D1-F1E0 GET a character
F1E1-F231 INPUT from any device
F232-F26D OUTPUT a character to any device
F26E Abort all files, and;
F284-F28C Restore normal I/O devices
F28D-F2A8 Find file table entry; set parameters from file table
F2A9-F300 Perform CLOSE
F301-F30E Test stop key
F30F-F314 Action stop key
F315-F31C Send message if direct mode
F31D-F321 Test if direct mode
F322-F3C1 Perform program loading
F3C2-F409 Perform LOAD
F40A-F43D Subroutines: Print SEARCHING... ; Print LOADING or VERIFYING
F43E-F45F Get Load or Save parameters
F460-F465 Get a byte parameter
F466-F493 Send program name to IEEE-488 bus
F494-F4B6 Find a specific tape header
F4B7-F4CD Perform VERIFY
F4CE-F50D Get parameters for OPEN, CLOSE
F50E-F515 Abort calling subroutine if end-of-line (default parameters)
F516-F520 Confirm comma, else send SYNTAX ERROR
F521-F5A5 Perform OPEN
F5A6-F5D9 Find any tape header
F5DA-F63B Write tape header

F63C-F655 Get start & end program addresses from tape header
F656-F66B Set cassette buffer address according to device number
F66C-F683 Set tape start & end addresses from buffer address
F684-F68C Perform CMD
F68D-F69D Set tape start & end addresses from Basic pointers
F69E-F728 Perform SAVE
F729-F76C Update TI and TI$, and copy STOP key to work area
F76D-F76F TI constant: limit of clock (24 hours)
F770-F7BB Set input device
F7BC-F805 Set output device
F806-F811 Advance tape buffer pointer (for INPUT#, GET#, and PRINT#)
F812-F834 Wait: PRESS PLAY ON TAPE#
F835-F846 Test if cassette button(s) pressed
F847-F854 Wait: PRESS PLAY & RECORD ON TAPE#
F855 Initiate tape read
F886-F8E5 Initiate tape write
F8E6-F8EF Test for I/O interrupt completion
F8F0-F8FF Test stop key
F900-F930 Set expected timing for next input bit from tape
F931 Interrupt entry: Read tape bits
FA57-FB75 Store received tape characters
FB76-FB7E Set tape read/write address back to starting point
FB7F-FB83 Flag I/O error into ST
FB84-FB92 Reset 8-count and flags for a new byte
FB93-FBAE Write a transition to cassette tape
FBAF-FC40 Write interrupt 2: write data to tape
FC41-FC7A Write interruot 1: write tape shorts (leader)
FC7B-FC95 Terminate tape: restore normal interrupt vector
FC96-FCA5 Set interrupt vector from table
FCA6-FCB3 Turn off cassette motors
FCB4-FCC5 Perform running checksum calculation
FCC6-FCD0 Advance read/write pointer
FCD1-FCFD Power-on reset entry point
FCFE-FD00 NMI interrupt entry point
FD01-FD10 Table of interrupt vectors
FD11-FFB0 Machine Language Monitor (MLM) - see Commodore documentation
FFB1-FFBF CBM copyright statement
*****Jump Table*****
FFC0 OPEN
FFC3 CLOSE
FFC6 Set input device
FFC9 Set output device
FFCC Restore default I/O devices
FFCF Input character
FFD2 Output character
FFD5 LOAD
FFD8 SAVE
FFDB VERIFY
FFDE SYS
FFE1 Test stop key
FFE4 Get character
FFE7 Abort all I/O activity
FFEA Clock update
FFF0-FFF9 unused
FFFA-FFFF Hardware vectors: NMI, Reset, Interrupt

Memory map: Original ROM to Upgrade ROM — Jim Butterfield

To identify a function of PET's original ROM, and/or convert it to the equivalent upgrade ROM location, use this table.

All addresses are given in hexadecimal.

OLD ADDRS	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
0000:	0000	0001	0002	000E	**	**	**	**
0008:	0011	0012	0200	0201	0202	0203	0204	0205
0010:	0206	0207	0208	0209	020A	020B	020C	020D
0018:	020E	020F	0210	0211	0212	0213	0214	0215
0020:	0216	0217	0218	0219	021A	021B	021C	021D
0028:	021E	021F	0220	0221	0222	0223	0224	0225
0030:	0226	0227	0228	0229	022A	022B	022C	022D
0038:	022E	022F	0230	0231	0232	0233	0234	0235
0040:	0236	0237	0238	0239	023A	023B	023C	023D
0048:	023E	023F	0240	0241	0242	0243	0244	0245
0050:	0246	0247	0248	0249	024A	024B	024C	024D
0058:	024E	024F	0003	0004	0005	0006	0007	0008
0060:	0009	000A	000B	000C	000D	0013	0014	0015
0068:	0016	0017	0018	0019	001A	001B	001C	001D
0070:	001E	001F	0020	0021	0022	0023	0024	0025
0078:	0026	0027	0028	0029	002A	002B	002C	002D
0080:	002E	002F	0030	0031	0032	0033	0034	0035
0088:	0036	0037	0038	0039	003A	003B	003C	003D
0090:	003E	003F	0040	0041	0042	0043	0044	0045
0098:	0046	0047	0048	0049	004A	004B	004C	004D
00A0:	004E	004F	0050	0051	0052	0053	0054	0055
00A8:	0056	0057	0058	0059	005A	005B	005C	005D
00B0:	005E	005F	0060	0061	0062	0063	0064	0065
00B8:	0066	0067	0068	0069	006A	006B	006C	006D
00C0:	006E	006F	0070	0071	0072	0073	0074	0075
00C8:	0076	0077	0078	0079	007A	007B	007C	007D
00D0:	007E	007F	0080	0081	0082	0083	0084	0085
00D8:	0086	0087	0088	0089	008A	008B	008C	**
00E0:	00C4	00C5	00C6	00C7	00C8	00C9	00CA	00CB
00E8:	00CC	00B4	00CD	00CE	00CF	00D0	00D1	00D2
00F0:	00D3	00D4	00D5	00D6	00D7	00D8	00D9	00FB
00F8:	00FC	00DA	00DB	00DC	00DD	00DE	00DF	**
0200:	008D	008E	008F	0097	0098	0099	009A	00F9
0208:	00FA	009B	009C	009D	0096	009E	009F	026F
0210:	0270	0271	0272	0273	0274	0275	0276	0277
0218:	0278	0090	0091	0092	0093	00A0	00A1	**
0220:	00A3	00A4	00A5	00A6	00A7	00A8	00A9	00AA
0228:	00AB	00E0	00E1	00E2	00E3	00E4	00E5	00E6
0230:	00E7	00E8	00E9	00EA	00EB	00EC	00ED	00EE
0238:	00EF	00F0	00F1	00F2	00F3	00F4	00F5	00F6
0240:	00F7	00F8	0251	0252	0253	.. etc.		
0260:	00AC	00AD	00AE	00AF	00B0	00B1	00B2	**
0268:	00B5	**	**	**	00B7	**	**	00B9
0270:	00BA	00BB	00BC	00BD	00BE	00BF	00C0	00C1

Memory locations for ROM upgrade on PET computers
Jim Butterfield, Toronto

0000-0002	0-2	USR Jump instruction
0003	3	Search character
0004	4	Scan-between-quotes flag
0005	5	Basic input buffer pointer; # subscripts
0006	6	Default DIM flag
0007	7	Type: FF=string, 00=numeric
0008	8	Type: 80=integer, 00=floating point
0009	9	DATA scan flag: LIST quote flag; memory flag
000A	10	Subscript flag; FNx flag
000B	11	0=input; 64=get; 152=read
000C	12	ATN sign flag; comparison evaluation flag
000D	13	input flag; suppress output if negative
000E	14	current I/O device for prompt-suppress
0011-0012	17-18	Basic integer address (for SYS, GOTO etc)
0013	19	Temporary string descriptor stack pointer
0014-0015	20-21	Last temporary string vector
0016-001E	22-30	Stack of descriptors for temporary strings
001F-0020	31-32	Pointer for number transfer
0021-0022	33-34	Misc. number pointer
0023-0027	35-39	Product staging area for multiplication
0028-0029	40-41	Pointer: Start-of-Basic memory
002A-002B	42-43	Pointer: End-of-Basic, Start-of-Variables
002C-002D	44-45	Pointer: End-of-Variables, Start-of-Arrays
002E-002F	46-47	Pointer: End-of-Arrays
0030-0031	48-49	Pointer: Bottom-of-Strings (moving down)
0032-0033	50-51	Utility string pointer
0034-0035	52-53	Pointer: Limit of Basic Memory
0036-0037	54-55	Current Basic line number
0038-0039	56-57	Previous Basic line number
003A-003B	58-59	Pointer to Basic statement (for CONT)
003C-003D	60-61	Line number, current DATA line
003E-003F	62-63	Pointer to current DATA item
0040-0041	64-65	Input vector
0042-0043	66-67	Current variable name
0044-0045	68-69	Current variable address
0046-0047	70-71	Variable pointer for FOR/NEXT
0048	72	Y save register; new-operator save
004A	74	Comparison symbol accumulator
004B-004C	75-76	Misc numeric work area
004D-0050	77-80	Work area: garbage yardstick
0051-0053	81-83	Jump vector for functions
0054-0058	84-88	Misc numeric storage area
0059-005D	89-93	Misc numeric storage area
005E-0063	94-99	Accumulator#1: E,M,M,M,M,S
0064	100	Series evaluation constant pointer
0065	101	Accumulator hi-order propogation word
0066-006B	102-107	Accumulator#2
006C	108	Sign comparison, primary vs. secondary
006D	109	low-order rounding byte for Acc#1
006E-006F	110-111	Cassette buffer length/Series pointer

Memory map, upgrade ROM, contd.

0070-0087	112-135	Subrtn: Get Basic Char; 77,78=pointer
0088-008C	136-140	RND storage and work area
008D-008F	141-143	Jiffy clock for TI and TI$
0090-0091	144-145	Hardware interrupt vector
0092-0093	146-147	Break interrupt vector
0094-0095	148-149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key depressed: 255=no key
0098	152	Shift key: 1 if depressed
0099-009A	153-154	Correction clock
009B	155	Keyswitch PIA: STOP and RVS flags
009C	156	Timing constant buffer
009D	157	Load=0, Verify=1
009E	158	# characters in keyboard buffer
009F	159	Screen reverse flag
00A0	160	IEEE-488 output flag: FF=character waiting
00A1	161	End-of-line-for-input pointer
00A3-00A4	163-164	Cursor log (row, column)
00A5	165	IEEE-488 output character buffer
00A6	166	Key image
00A7	167	0=flashing cursor, else no cursor
00A8	168	Countdown for cursor timing
00A9	169	Character under cursor
00AA	170	Cursor blink flag
00AB	171	EOT bit received
00AC	172	Input from screen/input from keyboard
00AD	173	X save flag
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00B1	177	Tape character parity
00B2	178	Byte received flag
00B4	180	Tape buffer character
00B5	181	Pointer in filename transfer
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Countdown for tape write
00BB	187	Tape buffer#1 count
00BC	188	Tape buffer#2 count
00BD	189	Write leader count; Read pass1/pass2
00BE	190	Write new byte; Read error flag
00BF	191	Write start bit; Read bit seq error
00C0	192	Pass 1 error log pointer
00C1	193	Pass 2 error correction pointer
00C2	194	0=Scan; 1-15=Count; $40=Load; $80=End
00C3	195	Checksum for read; Leader length for write
00C4-00C5	196-197	Pointer to screen line
00C6	198	Position of cursor on above line

Memory map, upgrade ROM, contd.

00C7-00C8	199-200	Utility pointer: tape buffer, scrolling
00C9-00CA	201-202	Tape end address/end of current program
00CB-00CC	203-204	Tape timing constants
00CD	205	00=direct cursor, else programmed cursor
00CE	206	Timer 1 enabled for tape read; 00=disabled
00CF	207	EOT signal received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current logical file number
00D3	211	Current secondary addrs, or R/W command
00D4	212	Current device number
00D5	213	Line length (40 or 80) for screen
00D6-00D7	214-215	Start of tape buffer, address
00D8	216	Line where cursor lives
00D9	217	Last key input; buffer checksum; bit buffer
00DA-00DB	218-219	File name pointer
00DC	220	Number of keyboard INSERTs outstanding
00DD	221	Write shift word/Receive input character
00DE	222	#blocks remaining to write/read
00DF	223	Serial word buffer
00E0-00F8	224-248	Screen line table: hi order address & line wrap
00F9	249	Cassette#1 status switch
00FA	250	Cassette#2 status switch
00FB-00FC	251-252	Tape start address
0100-010A	256-266	Binary to ASCII conversion area
0100-013E	256-318	Tape read error log for correction
0100-01FF	256-511	Processor stack area
0200-0250	512-592	Basic input buffer
0251-025A	593-602	Logical file number table
025B-0264	603-612	Device number table
0265-026E	613-622	Secondary address, or R/W cmd, table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 buffer
033A-03F9	826-1017	Tape#2 buffer
03FA-03FB	1018-1019	Vector for Machine Language Monitor
0400-7FFF	1024-32767	Available RAM including expansion
8000-8FFF	32768-36863	Video RAM
9000-BFFF	36864-49151	Available ROM expansion area
C000-E0F8	49152-57592	Microsoft Basic interpreter
E0F9-E7FF	57593-59391	Keyboard, Screen, Interrupt programs
E810-E813	59408-59411	PIA1 - Keyboard I/O
E820-E823	59424-59427	PIA2 - IEEE488 I/O
E840-E84F	59456-59471	VIA - I/O and Timers
F000-FFFF	61440-65535	Reset, tape, diagnostic monitor

PET MEMORY LOCATIONS — September 1978

0000-0002	0-2	USR Jump instruction
0003	3	Current I/O Device for prompt-suppress
0005	5	Cursor position for Input & Print
0008-0009	8-9	Integer address from Basic (for SYS, GOTO, etc.)
000A-0059	10-89	Basic input buffer; # of array subscripts
005A	90	Search character (usually ',' or end-of-line)
005B	91	Scan-between-quotes flag
005C	92	Basic input buffer pointer; number of subscripts
005D	93	First-character of array-name; default DIM flag
005E	94	Type: FF=string; 00=numeric
005F	95	Type: 80=integer; 00=floating point
0060	96	'DATA' scan flag; LIST quote flag; memory flag
0061	97	Subscript flag; FNx flag
0062	98	0=input, 64=get, 152=read (flag)
0063	99	flag for trigonometric signs/comparison evaluation flag
0064	100	input flag (suppress output if negative)
0065	101	variable pseudo-stack pointer
0066	102	fixed-point pseudo-stack pointer
0067	103	dummy value (0)
0068-0070	104-112	variable & pseudo-stack
0071-0072	113-114	pointer for number transfer
0073-0074	115-116	number pointer
0075-0078	117-120	product staging area for multiplication
007A-007B	122-123	start of basic pointer
007C-007D	124-125	end of basic/start of varibles pointer
007E-007F	126-127	end of variables/startof arrays
0080-0081	128-129	start of available space pointer
0082-0083	130-131	bottom of strings (moving down) pointer
0084-0085	132-133	top of strings (moving down) pointer
0086-0087	134-135	limit of Basic memory pointer
0088-0089	136-137	current program line number
008A-008B	138-139	previous line number
008C-008D	140-141	previous line address (for CONT)
008E-008F	142-143	line number of DATA line
0090-0091	144-145	memory address of DATA line
0092-0093	146-147	input vector (DATA etc.)
0094-0095	148-149	current variable name
0096-0097	150-151	current variable address
0098-0099	152-153	variable pointer for current FOR/NEXT
009A	154	Y save register; new operator save
009C	156	comparison symbol accumulator: <1 =2 >4
0090-00A1	157-161	number work area for SQR, etc.
00A2	162	pseudo-stack yardstick (3 or 7)
00A3-00A5	163-165	jump vector for functions
00A6-00AA	166-170	numeric store area
00AB-00AF	171-175	numeric store area
0080-0055	176-181	primary accumulator E,M,M,M,M,S
00B6	182	Taylor series constant counter
00B7	183	accumulator high-order propogation word
00B8-00BD	184-189	secondary accumulator
00BE	190	sign comparison, primary/secondary
00BF	191	low-order rounding byte for primary acc

September 1978

00C0-00C1	192-193	Cassette buffer length/Taylor constant pointer
00C2-0009	194-217	Subrtn: Get Basic Char; 89,8A=pointer
00DA-00DE	218-222	RND storage and work area
00E0-00E1	224-225	Pointer to screen cursor line
00E2	226	Position of cursor on line
00E3-00E4	227-228	Utility pointer; tape buffer,scrolling
00E5-00E6	229-230	End of current program/tape end address
00E7-00E8	231-232	Tape timing constants
00E9	233	Tape buffer character
00EA	234	Direct/programmed cursor; 00=direct
00EB	235	Tape read/verify flag
00EC	236	Tape write flag
00ED	237	
00EE	238	Number of % characters in file name
00EF	239	Logical file number
00F0	240	File command (from OPEN)
00F1	241	Device number
00F2	242	Maximum line length (40 or 80)
00F3-00F4	243-244	Tape buffr address (start of buffer)
00F5	245	Line where cursor lives
00F6	246	Last key pushed (ASCII); buffer checksum
00F7-00F8	247-248	Tape start address/tape pointer
00F9-00FA	249-250	File name pointer
00FB	251	Number of "insert" keys pushed
00FC	252	Serial bit shift word
00FD	253	# blocks remaining to write
00FE	254	Serial word buffer
0100-010A	256-266	Binary to ASCII conversion area
0100-013F	267-511	Stack area
0200-0202	512-514	TI and TI$ clock - jiffies
0203	515	Which key depressed: 255 = no key
0204	516	Shift key: 1 if depressed
0205-0206	517-518	Clock (unused?)
0207	519	Cassette 1 status switch
0208	520	Cassette 2 status switch
0209	521	Keyswitch PIA: STOP & RVS flags, etc.
020A	522	
020B	523	Load=0, Verify=1
020C	524	Status
020D	525	# characters in keyboard buffer
020E	526	Reverse flag
020F-0218	527-536	Keyboard buffer
0219-021A	537-538	Hardware interrupt vector
021B-021C	539-540	Break interrupt vector
021D	541	
021E	542	End-of-line-for -input pointer
0220-0221	544-545	Cursor log (row, column)
0222	546	PBD image for tape I/O
0223	547	Key image
0224	548	0=flashing cursor; else no cursor shows
0225	549	Cursor timing countdown
0226	550	Character under cursor
0227	551	Cursor blink flag
0228	552	Tape write
0229-02A1	553-577	Line address high / screen line wrap table

September 1978

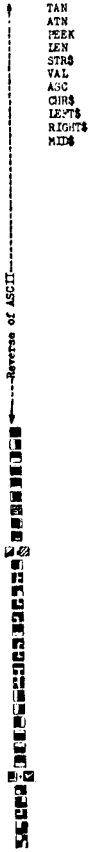
02A2-02AB	578-587	Logical numbers of open files
02AC-0255	588-597	Device numbers of open files
0256-025F	598-607	Command/Secondary address of open files
0260	608	Input from screen/input from keyboard
0261	609	X-save flag
0262	610	How many open files
0263	611	Input device, normally 0
0264	612	Output CMD device, normally 3
0265	613	Tape parity
0266	614	
0268	616	Pointer in filename transfer
026A	618	
026C	620	Serial bit count
026F	623	
0270	624	Tape write countdown
0271	625	Tape buffer #1 count
0272	626	Tape buffer #2 count
0273	627	Leader counter
0274	628	Flag for tape error
0275	629	0 if 1st ½-byte cntr not written
0276	630	2nd ½-byte cntr/tape error count
0277	631	
0278	632	Cassette read flag
0279	633	Checksum working word
027A-0339	634-825	Tape #1 buffer
033A-03F9	826-1017	Tape #2 buffer
0400-7FFF	1024-32767	Available RAM including expansion
8000-8FFF	32768-36863	Video RAM
9000-BFFF	36864-49151	Available ROM expansion area
C000-E077	49152-57463	Microsoft Basic
E078-E7F8	57464-59384	Keyboard/Screen/Interrupt monitor
E810	59408	PIA1 - Keyboard A register; (Direction with CRA2=1)
E811	59409	PIA1 - Keyboard A control
E812	59410	PIA1 - Keyboard B register; (Direction with CRB2=1)
E813	59411	PIA1 - Keyboard B control
E820	59424	PIA2 - IEEE A register; (Direction with CRA2=1)
E821	59425	PIA2 - IEEE A control
E822	59426	PIA2 - IEEE B register; (Direction with CRB2=1)
E823	59427	PIA2 - IEEE B control
E840	59456	VIA I/O register B
E841	59457	VIA I/O register A with handshake
E842-E843	59458-59459	VIA Data Direction regs, A and B
E844-E845	59460-59461	VIA Timer 1
E846-E847	59462-59463	VIA Timer 1 latch
E848-E849	59464-59465	VIA Timer 2
E84A	59466	VIA shift register
E84B	59467	ACS: T1.T1.T2.SR.SR.SR.PB.PA
E84C	59468	PCR: B2.B2.B2.B1.A2.A2.A2.A1
E84D-E84E	59469-59470	IFR, IER: T1.T2.CB1.BC2.SR.CA1.CA2
E84F	59471	I/O Register A without handshake
F000-FFFF	61440-65535	Reset/tape/diagnostic monitor

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
0	00		@	end-line	BRK	0
1	01		A		ORA(I,X)	1
2	02		B			2
3	03		C			3
4	04		D			4
5	05		E		ORA Z	5
6	06		F		ASL Z	6
7	07		G			7
8	08		H		PHP	8
9	09		I		ORA #	9
10	0A		J		ASL A	10
11	0B		K			11
12	0C		L			12
13	0D	car ret	M		ORA	13
14	0E		N		ASL	14
15	0F		O			15
16	10		P		BPL	16
17	11	cur down	Q		ORA(I),Y	17
18	12	reverse	R			18
19	13	cur home	S			19
20	14	delete	T			20
21	15		U		ORA Z,X	21
22	16		V		ASL Z,X	22
23	17		W			23
24	18		X		CLC	24
25	19		Y		ORA Y	25
26	1A		Z			26
27	1B		[			27
28	1C		\			28
29	1D	cur right	]		ORA X	29
30	1E		↑		ASL X	30
31	1F		←			31
32	20	space	space	space	JSR	32
33	21		!	!	AND(I,X)	33
34	22		"	"		34
35	23		#	#		35
36	24		$	$	BIT Z	36
37	25		%	%	AND Z	37
38	26		&	&	ROL Z	38
39	27		'	'		39
40	28		(	(	PLP	40
41	29		)	)	AND #	41
42	2A		*	*	ROL A	42
43	2B		+	+		43
44	2C		,	,	BIT	44
45	2D		-	-	AND	45
46	2E		.	.	ROL	46
47	2F		/	/		47
48	30		0	0	BMI	48
49	31		1	1	AND(I),Y	49
50	32		2	2		50
51	33		3	3		51
52	34		4	4		52
53	35		5	5	AND Z,X	53
54	36		6	6	ROL Z,X	54
55	37		7	7		55
56	38		8	8	SEC	56
57	39		9	9	AND Y	57
58	3A		:	:	CLI	58
59	3B		;	;		59
60	3C		<	<		60
61	3D		=	=	AND X	61
62	3E		>	>	ROL X	62
63	3F		?	?		63

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
64	40		@	@	RTI	64
65	41		A	A	EOR(I,X)	65
66	42		B	B		66
67	43		C	C		67
68	44		D	D		68
69	45		E	E	EOR Z	69
70	46		F	F	LSR Z	70
71	47		G	G		71
72	48		H	H	PHA	72
73	49		I	I	EOR #	73
74	4A		J	J	LSR A	74
75	4B		K	K		75
76	4C		L	L	JMP	76
77	4D		M	M	EOR	77
78	4E		N	N	LSR	78
79	4F		O	O		79
80	50		P	P	BVC	80
81	51		Q	Q	EOR(I),Y	81
82	52		R	R		82
83	53		S	S		83
84	54		T	T		84
85	55		U	U	EOR Z,X	85
86	56		V	V	LSR Z,X	86
87	57		W	W		87
88	58		X	X	CLI	88
89	59		Y	Y	EOR Y	89
90	5A		Z	Z		90
91	5B		[	[		91
92	5C		\	\		92
93	5D		]	]	EOR X	93
94	5E		↑	↑	LSR X	94
95	5F		←	←		95
96	60				RTS	96
97	61				ADC(I,X)	97
98	62					98
99	63					99
100	64					100
101	65				ADC Z	101
102	66				ROR Z	102
103	67					103
104	68				PLA	104
105	69				ADC #	105
106	6A				ROR A	106
107	6B					107
108	6C				JMP(I)	108
109	6D				ADC	109
110	6E				ROR	110
111	6F					111
112	70				BVS	112
113	71				ADC(I),X	113
114	72					114
115	73					115
116	74					116
117	75				ADC Z,X	117
118	76				ROR Z,X	118
119	77					119
120	78				SEI	120
121	79				ADC Y	121
122	7A					122
123	7B					123
124	7C					124
125	7D				ADC X	125
126	7E				ROR X	126
127	7F					127

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
128	80		r-@	END		128
129	81		r-A	FOR	STA(I,X)	129
130	82		r-B	NEXT		130
131	83		r-C	DATA		131
132	84		r-D	INPUT#	STY Z	132
133	85		r-E	INPUT	STA Z	133
134	86		r-F	DIM	STX Z	134
135	87		r-G	READ		135
136	88		r-H	LET	DEY	136
137	89		r-I	GOTO		137
138	8A		r-J	RUN	TXA	138
139	8B		r-K	IF		139
140	8C		r-L	RESTORE	STY	140
141	8D	car ret	r-M	GOSUB	STA	141
142	8E		r-N	RETURN	STX	142
143	8F		r-O	REM		143
144	90		r-P	STOP	BCC	144
145	91	cur up	r-Q	ON	STA(I),Y	145
146	92	rvs off	r-R	WAIT		146
147	93	clear	r-S	LOAD		147
148	94	insert	r-T	SAVE	STY Z,X	148
149	95		r-U	VERIFY	STA Z,X	149
150	96		r-V	DEF	STX Z,Y	150
151	97		r-W	POKE		151
152	98		r-X	PRINT#	TYA	152
153	99		r-Y	PRINT	STA Y	153
154	9A		r-Z	CONT	TXS	154
155	9B		r-[	LIST		155
156	9C		r-\	CLR		156
157	9D	cur left	r-]	CMD	STA X	157
158	9E		r-↑	SYS		158
159	9F		r-←	OPEN		159
160	A0		■	CLOSE	LDY #	160
161	A1		r-!	GET	LDA(I,X)	161
162	A2		r-"	NEW	LDX #	162
163	A3		r-#	TAB(		163
164	A4		r-$	TO	LDY Z	164
165	A5		r-%	FN	LDA Z	165
166	A6		r-&	SPC(	LDX Z	166
167	A7		r-'	THEN		167
168	A8		r-(	NOT	TAY	168
169	A9		r-)	STEP	LDA #	169
170	AA		r-*	+	TAX	170
171	AB		r-+	-		171
172	AC		r-,	*	LDY	172
173	AD		r--	/	LDA	173
174	AE		r-.	↑	LDX	174
175	AF		r-/	AND		175
176	B0		r-0	OR	BCS	176
177	B1		r-1	>	LDA(I),Y	177
178	B2		r-2	=		178
179	B3		r-3	<		179
180	B4		r-4	SGN	LDY Z,X	180
181	B5		r-5	INT	LDA Z,X	181
182	B6		r-6	ABS	LDX Z,Y	182
183	B7		r-7	USR		183
184	B8		r-8	FRE	CLV	184
185	B9		r-9	POS	LDA Y	185
186	BA		r-:	SQR	TSX	186
187	BB		r-;	RND		187
188	BC		r-<	LOG	LDY X	188
189	BD		r-=	EXP	LDA X	189
190	BE		r->	COS	LDX Y	190
191	BF		r-?	SIN		191

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
192	C0			TAN	CPY #	192
193	C1		,a	ATN	CMP(I),X	193
194	C2		,b	PEEK		194
195	C3		,c	LEN		195
196	C4		,d	STR$	CPY Z	196
197	C5		,e	VAL	CMP Z	197
198	C6		,f	ASC	DEC Z	198
199	C7		,g	CHR$		199
200	C8		,h	LEFT$	INY	200
201	C9		,i	RIGHT$	CMP #	201
202	CA		,j	MID$	DEX	202
203	CB		,k			203
204	CC		,l		CPY	204
205	CD		,m		CMP	205
206	CE		,n		DEC	206
207	CF		,o			207
208	D0		,p		BNE	208
209	D1		,q		CMP(I),Y	209
210	D2		,r			210
211	D3		,s			211
212	D4		,t			212
213	D5		,u		CMP Z,X	213
214	D6		,v		DEC Z,X	214
215	D7		,w			215
216	D8		,x		CLD	216
217	D9		,y		CMP Y	217
218	DA		,z			218
219	DB					219
220	DC					220
221	DD				CMP X	221
222	DE				DEC X	222
223	DF					223
224	E0				CPX #	224
225	E1				SBC(I),X	225
226	E2					226
227	E3					227
228	E4				CPX Z	228
229	E5				SBC Z	229
230	E6				INC Z	230
231	E7					231
232	E8				INX	232
233	E9				SBC #	233
234	EA				NOP	234
235	EB					235
236	EC				CPX	236
237	ED				SBC	237
238	EE				INC	238
239	EF					239
240	F0				BEQ	240
241	F1				SBC(I),Y	241
242	F2					242
243	F3					243
244	F4					244
245	F5				SBC Z,X	245
246	F6				INC Z,X	246
247	F7					247
248	F8				SED	248
249	F9				SBC Y	249
250	FA					250
251	FB					251
252	FC					252
253	FD				SBC X	253
254	FE				INC X	254
255	FF					255



32768

2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799	2800	2801	2802	2803	2804	2805	2806	2807
2808	2809	2810	2811	2812	2813	2814	2815	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879	2880	2881	2882	2883	2884	2885	2886	2887
2888	2889	2890	2891	2892	2893	2894	2895	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959	2960	2961	2962	2963	2964	2965	2966	2967
2968	2969	2970	2971	2972	2973	2974	2975	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039	3040	3041	3042	3043	3044	3045	3046	3047
3048	3049	3050	3051	3052	3053	3054	3055	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119	3120	3121	3122	3123	3124	3125	3126	3127
3128	3129	3130	3131	3132	3133	3134	3135	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199	3200	3201	3202	3203	3204	3205	3206	3207
3208	3209	3210	3211	3212	3213	3214	3215	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279	3280	3281	3282	3283	3284	3285	3286	3287
3288	3289	3290	3291	3292	3293	3294	3295	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359	3360	3361	3362	3363	3364	3365	3366	3367
3368	3369	3370	3371	3372	3373	3374	3375	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439	3440	3441	3442	3443	3444	3445	3446	3447
3448	3449	3450	3451	3452	3453	3454	3455	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519	3520	3521	3522	3523	3524	3525	3526	3527
3528	3529	3530	3531	3532	3533	3534	3535	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599	3600	3601	3602	3603	3604	3605	3606	3607
3608	3609	3610	3611	3612	3613	3614	3615	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679	3680	3681	3682	3683	3684	3685	3686	3687
3688	3689	3690	3691	3692	3693	3694	3695	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759	3760	3761	3762	3763	3764	3765	3766	3767

33767

The printing mode (standard or lower case) is set by POKEing an address. So as not to disturb any of the other bits in the peripheral control register a safe way to set the lower case mode would be: POKE 59468,PEEK(59468) OR 14 and reset it to standard mode with POKE 59468, PEEK(59468) AND 253 OR 12.

Standard Mode: Location 59468 = XXXX110X

	OFF	RVS	CHR$
@	64	192	192
	0	128	64
A	65	193	193
	1	129	65
B	66	194	194
	2	130	66
C	67	195	195
	3	131	67
D	68	196	196
	4	132	68
E	69	197	197
	5	133	69
F	70	198	198
	6	134	70
G	71	199	199
	7	135	71
H	72	200	200
	8	136	72
I	73	201	201
	9	137	73
J	74	202	202
	10	138	74
K	75	203	203
	11	139	75
L	76	204	204
	12	140	76
M	77	205	205
	13	141	77
N	78	206	206
	14	142	78
O	79	207	207
	15	143	79
RETURN			141
			13

	OFF	RVS	CHR$
P	80	208	208
	16	144	80
Q	81	209	209
	17	145	81
R	82	210	210
	18	146	82
S	83	211	211
	19	147	83
T	84	212	212
	20	148	84
U	85	213	213
	21	149	85
V	86	214	214
	22	150	86
W	87	215	215
	23	151	87
X	88	216	216
	24	152	88
Y	89	217	217
	25	153	89
Z	90	218	218
	26	154	90
[	91	219	219
	27	155	91
\	92	220	220
	28	156	92
]	93	221	221
	29	157	93
↑	94	222	222
	30	158	94
←	95	223	223
	31	159	95
CLR HOME			147
			19
OFF RVS On			146
			18

	OFF	RVS	CHR$
SP	96	224	160
	32	160	32
!	97	225	161
	33	161	33
"	98	226	162
	34	162	34
#	99	227	163
	35	163	35
$	100	228	164
	36	164	36
%	101	229	165
	37	165	37
&	102	230	166
	38	166	38
'	103	231	167
	39	167	39
(	104	232	168
	40	168	40
)	105	233	169
	41	169	41
*	106	234	170
	42	170	42
+	107	235	171
	43	171	43
,	108	236	172
	44	172	44
-	109	237	173
	45	173	45
.	110	238	174
	46	174	46
/	111	239	175
	47	175	47
CTRL			145
			17
Cursor			157
			29

	OFF	RVS	CHR$
0	112	240	176
	48	176	48
1	113	241	177
	49	177	49
2	114	242	178
	50	178	50
3	115	243	179
	51	179	51
4	116	244	180
	52	180	52
5	117	245	181
	53	181	53
6	118	246	182
	54	182	54
7	119	247	183
	55	183	55
8	120	248	184
	56	184	56
9	121	249	185
	57	185	57
:	122	250	186
	58	186	58
;	123	251	187
	59	187	59
<	124	252	188
	60	188	60
=	125	253	189
	61	189	61
>	126	254	190
	62	190	62
?	127	255	191
	63	191	63
INST DEL			148
			20
RUN STOP			131
			3

Lower Case Mode: Location 59468 / XXXX110X, Same Except 193 to 218 Prints as Lower Case a to z Plus Different Graphics:

105	233	169	
122	250	186	
94	222	222	
95	223	223	

0: End of line
1-31: unused
32: space
33: !
34: "
35: #
36: $
37: %
38: &
39: '
40: (
41:)
42: *
43: +
44: ,
45: -
46: .
47: /
48: 0
49: 1
50: 2
51: 3
52: 4
53: 5
54: 6
55: 7
56: 8
57: 9
58: :
59: ;
60: <
61: =
62: >
63: ?
64: @
65: A
66: B
67: C
68: D
69: E
70: F
71: G
72: H
73: I
74: J
75: K
76: L
77: M
78: N
79: O
80: P
81: Q
82: R
83: S
84: T
85: U
86: V
87: W
88: X
89: Y
90: Z
91: [
92: \
93:]
94: ↑
95: ←
96: space
97: |
98: "
99: #
100: $
101: %
102: &
103: '
104: (
105:)
106: *
107: +
108: ,
109: -
110: .
111: /
112: 0
113: 1
114: 2
115: 3
116: 4
117: 5
118: 6
119: 7
120: 8
121: 9
122: :
123: ;
124: <
125: =
126: >
127: ?
128: END
129: FOR
130: NEXT
131: DATA
132: INPUT#
133: INPUT
134: DIM
135: READ
136: LET
137: GOTO
138: RUN
139: IF
140: RESTORE
141: GOSUB
142: RETURN
143: REM
144: STOP
145: ON
146: WAIT
147: LOAD
148: SAVE
149: VERIFY
150: DEF
151: POKE
152: PRINT#
153: PRINT
154: CONT
155: LIST
156: CLR
157: CMD
158: SYS
159: OPEN
160: CLOSE
161: GET
162: NEW
163: TAB(
164: TO
165: FN
166: SPC(
167: THEN
168: NOT
169: STEP
170: +
171: -
172: *
173: /
174: ↑
175: AND
176: OR
177: >
178: =
179: <
180: SGN
181: INT
182: ABS
183: USR
184: FRE
185: POS
186: SQR
187: RND
188: LOG
189: EXP
190: COS
191: SIN
192: TAN
193: ATN
194: PEEK
195: LEN
196: STR$
197: VAL
198: ASC
199: CHR$
200: LEFT$
201: RIGHT$
202: MID$
203-254: unused
255: π

Basic Commands and Statements

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
CLR	CLR	Sets variables to zero or null.
CMD	CMD D	Keep IEEE device D open to monitor bus.
CONT	CONT	Continue program execution after a STOP command. No program changes permitted.
GOTO	GOTO L	Continue program execution at line L after a STOP command. Program changes are permitted.
FRE	PRINT FRE (0)	Returns number of bytes of available memory.

Basic Commands and Statements (Continued)

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
DATA	10 DATA 1,2,3,4	Specifies data to be read from left to right.
	20 DATA TOM,SUE	Alphabetics do not need to be enclosed in quotes.
	30 DATA "TOM DOE"	If strings contain spaces, commas, colons, or graphic characters, the string must be enclosed in quotes.
DIM	10 DIM A(n)	Specifies maximum number of elements in an array or matrix.
	20 DIM A(n,m,o,p)	Specifies maximum number of dimensions in an array.
	30 DIM A(n),B(m)	Number of arrays limited by memory.
	40 DIM A(N)	May be dimensioned dynamically.
	50 DIM A$(n)	Strings may be dimensioned.
END	999 END	Terminates program execution.
GET	10 GET C	Accepts single character from keyboard.
	20 GET C$	Accepts single string character from keyboard.
	30 GET #D,C	Accepts single character from specified logical file.
	40 GET #D,C$	Accepts specified single string character from logical file.
INPUT	10 INPUT A	Accepts value of A from keyboard.
	20 INPUT A$	Accepts value of string variable A from keyboard. The string does not have to be enclosed in quotes.
	30 INPUT A,A$,B,B$	Accepts specified values from keyboard.
	40 INPUT #D,A	Accepts value of A from logical file D.
	50 INPUT #D,A$	Accepts specified string from logical file D.
	60 INPUT #D,A,A$,B,B$	Accepts specified values and strings from logical file D. Strings do not have to be enclosed in quotes.
LOAD	10 LOAD	Loads next encountered program or file, on built-in tape unit, into PET's memory.
	20 LOAD "NAME"	Loads program or file NAME into memory from built-in tape unit.
	30 LOAD "NAME",D	Loads specified file NAME from device D.
OPEN	10 OPEN A	Opens logical file A for read only from built-in tape unit.
	20 OPEN A,D	Opens logical file A for read only from device D.
	30 OPEN A,D,C	Opens logical file A for command C from device D.
	40 OPEN A,D,C,"NAME"	Opens logical file A on device D. If device D accepts formatted files, file NAME is positioned for command.
POS	10 PRINT POS(0)	Prints next available print position (position of cursor on screen).
PRINT	10 PRINT A	Prints value of A on display screen.
	20 PRINT A$	Prints specified string on screen.
	30 PRINT A,A$	Prints specified values or strings on screen, beginning in next available print position (pre-TABbed positions are in columns 10,20,30,40, etc.).

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
LIST	LIST	Lists current program.
	LIST -L	Lists current program through line L.
	LIST L-M	Lists lines L through M of current program.
	LIST L-	Lists current program from line L to end.
LOAD	LOAD	Loads next encountered program from built-in tape unit.
	LOAD "NAME"	Loads file NAME from built-in tape unit.
	LOAD "NAME," D	Loads file NAME from device D.
NEW	NEW	Deletes current program from memory, sets variables to zero.
PEEK	PEEK(A)	Returns byte value from address A.
POKE	POKE A,B	Loads byte B into address A.
PRINT	PRINT A	Prints value of A on display screen.
	PRINT A$	Prints specified string on screen.
	PRINT #D,A	Prints value of A on device D.
	PRINT #D,A$	Prints specified string on device D.
RUN	RUN	Begins execution of program at lowest line number.
	RUN L	Begins execution of program at line L.
SAVE	SAVE	Saves current program on built-in tape unit.
	SAVE "NAME"	Saves current file or program NAME on built-in tape unit.
	SAVE "NAME," D	Saves current program or file NAME on device D.
	SAVE "NAME," D,C	Saves file NAME on device D. C specifies EOF or EOT.
STOP	STOP	Stops program execution.
SYS	SYS(X)	Complete control of PET is transferred to a subsystem at decimal address contained in the argument.
TI$	TI$="HHMMSS"	Sets PET's internal clock to real time.
	PRINT TI	Displays number of 'jiffies' since PET was powered up or clock was zeroed. (A jiffy = 1/60 of a second.)
USR	USR(X)	Transfers program control to a program whose address is at locations 1 and 2. X is a parameter passed to and from the machine language program.
WAIT	WAIT A,B,C	Stops execution of BASIC until contents of A, ANDed with B and exclusive ORed with C, is not equal to zero. C is optional and defaults to zero.
CLOSE	10 CLOSE N	Closes logical file N.

Basic Commands and Statements (Continued)

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
GOTO	10 GOTO L	Transfers control (jumps) to specified line, skipping over intervening lines.
GOSUB	10 GOSUB L	Begins execution of a subroutine which begins on a specified line.
ON . . . GOTO	10 ON A GOTO L,M,N	Transfers control to specified line (in this example, L,M, or N, depending on value of index A.
ON . . . GOSUB	10 ON A GOSUB L,M,N	Begins execution of subroutine which begins on line L,M, or N, depending on the value of index A.
RETURN	9990 RETURN	Subroutine exit; transfers control to the statement following most recent GOSUB directing transfer to the subroutine.

String Functions

FUNCTION	EXAMPLE	PURPOSE
ASC	10 A=ASC("XYZ")	Returns integer value corresponding to ASCII code of first character in string.
CHR$	10 A$=CHR$(N)	Returns character corresponding to ASCII code number.
LEFT$	10 ?LEFT$(X$,A)	Returns leftmost A characters from string.
LEN	10 ?LEN(X$)	Returns length of string.
MID$	10 ?MID$(X$,A,B)	Returns B characters from string, starting with the Ath character.
RIGHT$	10 ?RIGHT$(X$,A)	Returns rightmost A characters from string.
STR$	10 A$=STR$(A)	Returns string representation of number.
VAL	10 A=VAL(A$) 20 A=VAL("A")	Returns numeric representation of string. If string not numeric, returns "Ø".

ASC, LEN and VAL functions return numerical results. They may be used as part of an expression. Assignment statements are used here for examples only; other statement types may be used.

Arithmetic Functions

FUNCTION	EXAMPLE	PURPOSE
ABS	10 C=ABS(A)	Returns magnitude of argument without regard to sign.
ATN	10 C=ATN(A)	Returns arctangent of argument. C will be expressed in radians.
COS	10 C=COS(A)	Returns cosine of argument. A must be expressed in radians.
DEF FN	10 DEF FNA(B)=C*D	Allows user to define a function. Function label A must be a single letter; argument B is a dummy.
EXP	10 C=EXP(A)	Returns constant 'e' raised to power of the argument. In this example, e^A.
INT	10 C=INT(A)	Returns largest integer less than or equal to argument.
LOG	10 C=LOG(A)	Returns natural logarithm of argument. Argument must be greater than or equal to zero.
RND	10 C=RND(A)	Generates a random number between zero and one. If A is less than 0, the same random number is produced in each call to RND. If A = 0, the same sequence of random numbers is generated each time RND is called. If A is greater than 0, a new sequence is produced for each call to RND.
SGN	10 C=SGN(A)	Returns -1 if argument is negative, returns 0 if argument is zero, and returns +1 if argument is positive.
SIN	10 C=SIN(A)	Returns sine or argument. A must be expressed in radians.
SQR	10 C=SQR(A)	Returns square root of argument.
TAN	10 C=TAN(A)	Returns tangent of argument. A must be expressed in radians.

Arithmetic Operators

SYMBOL	EXAMPLE	PURPOSE
=	10 A=B 20 LET A=B	Assigns a value to a variable. Let is optional.
↑	30 PRINT A↑2	Exponentiation; in example, A^2.
/	35 C=A/8	Division
*	40 C=A*8	Multiplication
+	50 C=A+8	Addition
-	60 C=A-8	Subtraction
=	10 IF A=B THEN PRINT C	A 'equals' B.
<>	10 IF A<>B THEN C=4	A 'does not equal' B.
<	10 IF A<B THEN C$="X"	A 'is less than' B.
>	10 IF A>B THEN C$=D$+E$	A 'is greater than' B.
<=	10 IF A<=B THEN C=20	A 'is less than or equal to' B.

Arithmetic Operators (Continued)

SYMBOL	EXAMPLE	PURPOSE
>=	10 IF A>=B THEN C=D-1	A 'is greater than or equal to' B.
AND	10 IF A AND B THEN C=0	A and B must BOTH be true for statement 10 to be true.
OR	20 IF A OR B THEN C=90	A must be true or B must be true for statement 20 to be true.
NOT	30 IF NOT A THEN PRINT C	Expression is true if A is false.

**NOTE: The numerical values used in the evaluation of logical comparisons are: 'TRUE' is any non-zero number and 'FALSE' is zero.

Special Symbols, Commands and Statements

SYMBOLS, COMMANDS, STATEMENTS	EXAMPLE	PURPOSE
:	10A=1:B=2:C=3	Allows multiple statements on a line.
;	10PRINT A;B	Allows same line printing. Elements are separated by 3 spaces.
	20PRINT A$;B$	Allows same line printing. String elements are concatenated.
,	10PRINT A,B	Allows same line printing. Elements are separated and printed in pre-TABbed print positions (columns 10,20,30, etc.)
,	LOAD "NAME," D	Separates elements in LOAD, SAVE, OPEN, and VERIFY.
?	10?A	Abbreviation for PRINT. Stores as one character; lists as word PRINT.
$	10A$="ABCDEFG"	String identifier.
%	10A%=INT(X)	Integer identifier.
" "	10A$="ABCDEF"	String enclosures.
carriage return		Must follow every command, statement, or data entry; causes cursor to return to leftmost position on next lowest line. Signals "END OF INPUT LINE."
π		Value of Pi: 3.1415927.

I/O Commands

SYMBOL	COMMAND	PURPOSE
L=1-255 C=0: READ C=1: WRITE C=2: WRITE AND PUT EOT at end of file. D=1 CASSETTE D=2 2ND CASSETTE D=4-15 IEEE BUSS	OPEN L,D,C	Note: PET will not read past an EOT (end of tape) marker.

Table Status Word (ST) values correlated with tape cassette, unit and IEEE bus read/write errors.

ST Bit Position	ST Numeric Value	Cassette Read	IEEE R/W	Tape Verify + Load
0	1		Time out on write	
1	2		Time out on read	
2	4	Short block		Short block
3	8	Long block		Long block
4	16	Unrecoverable read error		Any mismatch
5	32	Checksum error		Checksum error
6	64	End of file	EOI line	
7	-128	End of tape	Device not present	End of tape

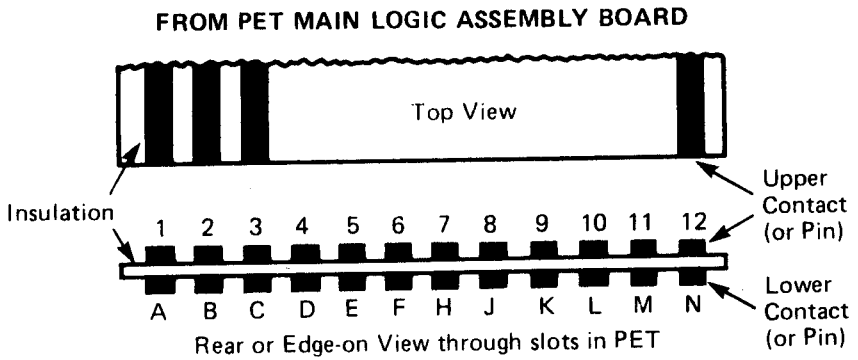


Figure 1-2. Simplified views of edge connectors J1 and J2 to illustrate contact identification convention.

Table: Parallel user port information.
PET pin identification characters, the corresponding signal labels and their descriptions. →

Pin Identification Character	Signal Label	Signal Description
1	Ground	Digital ground.
2	T.V. Video	Video output used for external display, used in diagnostic routine for verifying the video circuit to the display board.
3	IEEE-SRQ	Direct connection to the SRQ signal on the IEEE-488 port. It is used in verifying operation of the SRQ in the diagnostic routine.
4	IEEE-EOI	Direct connection to the EOI signal on the IEEE-488 port. It is used in verifying operation of the EOI in the diagnostic routine.
5	Diagnostic Sense	When this pin is held low during power up the PET software jumps to the diagnostic routine, rather than the BASIC routine.
6	Tape #1 READ	Used with the diagnostic routine to verify cassette tape #1 read function.
7	Tape #2 READ	Used with the diagnostic routine to verify cassette tape #2 read function.
8	Tape Write	Used with the diagnostic routine to verify operation of the WRITE function of both cassette ports.
9	T.V. Vertical	T.V. vertical sync signal verified in diagnostic. May be used for external TV display.
10	T.V. Horizontal	T.V. horizontal signal verified in diagnostic may be used for TV display.
11, 12	GND	Digital ground.
A	GND	Digital ground.
B	CA1	Standard edge sensitive input of 6522VIA.
C	PA0	Input/output lines to peripherals, and can be programmed independently of each other for input or output.
D	PA1	
E	PA2	
F	PA3	
H	PA4	
J	PA5	
K	PA6	
L	PA7	
M	CB2	Special I/O pin of VIA.
N	GND	Digital ground.

Table: Second cassette interface port.
PET pin identification characters, labels and associated descriptions.

Note A-1, B-2, etc., imply a pin A to pin1, pin B to pin 2, connection.
In some special units, pins 1 through 6 were not connected.

Pin Identification Characters	Label	Description
A-1	GND	Digital ground.
B-2	+5	Positive 5 volts to operate cassette circuitry only.
C-3	Motor	Computer controlled positive 6 volts for cassette motor.
D-4	Read	Read line from cassette.
E-5	Write	Write line to cassette.
F-6	Sense	Monitors closure of mechanical switch on cassette when any button is pressed.

Table: IEEE standard connectors

Connector Manufacturer	Identifier	Description
Cinch	5710240	Solder-plug
Cinch	5720240	Solder-receptacle
Amp	552301-1	Insulation displacement plug
Amp	552305-1	Insulation displacement receptacle

Table: A selection of suitable receptacles for connecting with the PET second cassette edge connector J3.

Manufacturer	Identifier
Sylvania	6AJ07-6-1A1-01
Viking	2KH6/1AB5
Viking	2KH6/9AB5
Viking	2KH6/21AB5
Amp	530692-1
Sullins	ESM6-SREH
Cinch	250-06-90-170

Table: Receptacles recommended for PET IEEE-488 connectors or parallel user port.

Manufacturer	Part Number
Cinch	251-12-90-160
Sylvania	6AG01-12-1A1-01
Amp	530657-3
Amp	530658-3
Amp	530654-3

Table: IEEE-488 bus signal.

Bus Group	Signal Abbrev.	Name	Functional Description
Manager	ATN	Attention	The PET (controller) sets this signal low while it is sending commands on the data bus. When ATN is low, only peripheral addresses and control messages are on the data bus. When ATN is high, only preveiously assigned devices can transfer data.
Transfer	DAV	Data Valid	When DAV is low, this signifies that data is valid on data bus.
Manager	EOI	End or Identify	When the last byte of data is being transferred, the talker has the option of setting EOI low. The PET always sets EOI low while the last data byte is being transferred from the PET.
Manager	IFC	Interface Clear	The PET sends its internal reset signal as IFC low (true) to initialize all devices to the idle state. When PET is switched on or reset, IFC goes low for about 100 milliseconds..
Transfer	NDAC	Data Not Accepted	This signal is held low (true) by the listener while reading. When the data byte has been read, the listener sets NDAC high. This signals the talker that data has been accepted.
Transfer	NRFD	Not Ready for Data	When NRFD is low (true), one or more listeners are not ready for the *next* byte of data. When all devices are ready, NRFD goes high.
Manager	SRQ	Service Request	Not implemented in BASIC, but available to the PET user.
Manager	REN	Remote Enable	REN is held low by the bus controller. The PET has a pin grounded that keeps REN permanently low.
Data	DI01-8	Data input/ output lines 1 through 8	These signals represent the bits of information on the data bus. When a DI0 signal is low, it represents 1 and when high Ø.
General	GND	Ground	Ground connections: There are six control and management signal ground returns, one data signal ground return and one chassis shield ground lead.

PET Pin Characters	Standard IEEE Connector Pin Numbers	IEEE Signal Mnemonic	Signal Definition/Label
Upper Pins			
1	1	DI01	Data input/output line #1
2	2	DI02	Data input/output line #2
3	3	DI03	Data input/output line #3
4	4	DI04	Data input/output line #4
5	5	EOI	End or identify
6	6	DAV	Data valid
7	7	NRFD	Not ready for data
8	8	NDAC	Data not accepted
9	9	IFC	Interface clear
10	10	SRQ	Service request
11	11	ATN	Attention
12	12	GND	Chassis ground and IEEE cable shield drain wire
Lower Pins			
A	13	DI05	Data input/output line #5
B	14	DI06	Data input/output line #6
C	15	DI07	Data input/output line #7
D	16	DI08	Data input/output line #8
E	17	REN	Remote enable
F	18	GND	DAV ground
H	19	GND	NRFD ground
J	20	GND	NDAC ground
K	21	GND	IFC ground
L	22	GND	SRQ ground
M	23	GND	ATN ground
N	24	GND	Data ground (DI01-8)

Original ROM Board

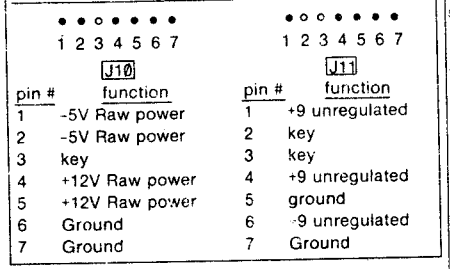
Table: **Memory expansion connector. PET pin numbers. Line labels and line descriptions.**

Side A Pin Numbers	Line Labels	Line Description
A1	BAØ	Address bit Ø, used for memory expansion. Buffered.
A2	BA1	Address bit 1, used for memory expansion. Buffered.
A3	BA2	Address bit 2, used for memory expansion. Buffered.
A4	BA3	Address bit 3, used for memory expansion. Buffered.
A5	BA4	Address bit 4, used for memory expansion. Buffered.
A6	BA5	Address bit 5, used for memory expansion. Buffered.
A7	BA6	Address bit 6, used for memory expansion. Buffered.
A8	BA7	Address bit 7, used for memory expansion. Buffered.
A9	BA8	Address bit 8, used for memory expansion. Buffered.
A1Ø	BA9	Address bit 9, used for memory expansion. Buffered.
A11	BA1Ø	Address bit 1Ø, used for memory expansion. Buffered.
A12	BA11	Address bit 11, used for memory expansion. Buffered.
A13	NC	No connection.
A14	NC	No connection.
A15	NC	No connection.
A16	$\overline{\text{SEL 1}}$	4K byte page address select for memory locations 1ØØØ-1FFF.
A17	$\overline{\text{SEL 2}}$	4K byte page address select for memory locations 2ØØØ-2FFF.
A18	$\overline{\text{SEL 3}}$	4K byte page address select for memory locations 3ØØØ-3FFF.
A19	$\overline{\text{SEL 4}}$	4K byte page address select for memory locations 4ØØØ-4FFF.
A2Ø	$\overline{\text{SEL 5}}$	4K byte page address select for memory locations 5ØØØ-5FFF.
A21	$\overline{\text{SEL 6}}$	4K byte page address select for memory locations 6ØØØ-6FFF.
A22	$\overline{\text{SEL 7}}$	4K byte page address select for memory locations 7ØØØ-7FFF.
A23	$\overline{\text{SEL 9}}$	4K byte page address select for memory locations 9ØØØ-9FFF.
A24	$\overline{\text{SEL A}}$	4K byte page address select for memory locations AØØØ-AFFF.
A25	$\overline{\text{SEL B}}$	4K byte page address select for memory locations BØØØ-BFFF.
A26	NC	No connection.
A27	$\overline{\text{RES}}$	Reset for 6502 microprocessor. Note: connected to 74LS00 output.
A28	$\overline{\text{IRQ}}$	Interrupt request line to the microprocessor.
A29	BØ2	Buffered phase 2 clock.
A3Ø	R/W	Buffered read/write from 6502 microprocessor.
A31	NC	No connection.
A32	NC	No connection.
A33	BDØ	Data bit Ø. Buffered.
A34	BD1	Data bit 1. Buffered.
A35	BD2	Data bit 2. Buffered.
A36	BD3	Data bit 3. Buffered.
A37	BD4	Data bit 4. Buffered.
A38	BD5	Data bit 5. Buffered.
A39	BD6	Data bit 6. Buffered.
A4Ø	BD7	Data bit 7. Buffered.

Side B (top) pins are Ground Returns for corresponding Side A (bottom) pins.

Upgrade ROM Board

Daughter board power connections



J10

pin #	function
1	-5V Raw power
2	-5V Raw power
3	key
4	+12V Raw power
5	+12V Raw power
6	Ground
7	Ground

J11

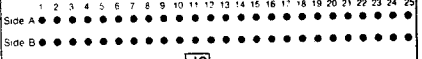
pin #	function
1	+9 unregulated
2	key
3	key
4	+9 unregulated
5	ground
6	-9 unregulated
7	Ground

Manufacturer	contact grid	identifier
Spectra-strip	2x7	8Ø2-1Ø4
Spectra-strip	2x7	8Ø2-114
* Spectra-strip	2x25	8Ø2-Ø5Ø
* Spectra-strip	2x25	8Ø2-15Ø
Circuit-Assembly	2x7	CA-14-IDSC
Circuit-Assembly	2x25	CA-50-IDSC

Table 7.12. A selection of suitable receptacles for connecting with PET daughter board pin connectors J4, J9, J1Ø, and J11

* Available at: Deskin Sales
77 - D Steelcase Rd. W.
Toronto, Ont.
(416) 495 - 1412

Memory expansion bus



J9

pin #	function	pin #	function
Side A1	ground	14	BA12
2	BAØ	15	BA13
3	BA1	16	BA14
4	BA2	17	BA15
5	BA3	18	SYNC
6	BA4	19	$\overline{IRQ}$
7	BA5	20	Memory Management
8	BA6	21	BØ2
9	BA7	22	BR/W
10	BA8	23	$\overline{BR/W}$
11	BA9	24	$\overline{DMA}$
12	BA1Ø	25	ground
13	BA11	Side B1-25	ground

J4

pin #	function	pin #	function
Side A1	ground	14	$\overline{SEL\ 6}$
2	BDØ	15	$\overline{SEL\ 7}$
3	BD1	16	$\overline{SEL\ 8}$
4	BD2	17	$\overline{SEL\ 9}$
5	BD3	18	$\overline{SEL\ A}$
6	BD4	19	$\overline{SEL\ B}$
7	BD5	20	$\overline{CAS}$
8	BD6	21	$\overline{RAS}$
9	BD7	22	$\overline{RES}$
10	$\overline{SEL\ 2}$	23	$\overline{RDY}$
11	$\overline{SEL\ 3}$	24	$\overline{NMI}$
12	$\overline{SEL\ 4}$	25	ground
13	$\overline{SEL\ 5}$	Side B1-25	ground

Connector Pin Numbers	Line Labels	Line Description
J9-1	GND	Logic Ground
J9-2	BAØ	Address bit Ø, used for memory expansion. Buffered.
J9-3	BA1	Address bit 1, used for memory expansion. Buffered.
J9-4	BA2	Address bit 2, used for memory expansion. Buffered.
J9-5	BA3	Address bit 3, used for memory expansion. Buffered.
J9-6	BA4	Address bit 4, used for memory expansion. Buffered.
J9-7	BA5	Address bit 5, used for memory expansion. Buffered.
J9-8	BA6	Address bit 6, used for memory expansion. Buffered.
J9-9	BA7	Address bit 7, used for memory expansion. Buffered.
J9-25	GND	Logic Ground.
J9-10	BA8	Address bit 8, used for memory expansion. Buffered.
J9-11	BA9	Address bit 9, used for memory expansion. Buffered.
J9-12	BA10	Address bit 1Ø, used for memory expansion. Buffered.
J9-13	BA11	Address bit 11, used for memory expansion. Buffered.
J9-14	BA12	Address bit 12, used for memory expansion. Buffered.
J9-15	BA13	Address bit 13, used for memory expansion. Buffered.
J9-16	BA14	Address bit 14, used for memory expansion. Buffered.
J9-17	BA15	Address bit 15, used for memory expansion. Buffered.
J9-19	$\overline{IRQ}$	Interrupt request line to the microprocessor.
J9-21	$\overline{BØ2}$	Buffered phase 2 clock.
J9-22	BR/W	Buffered read/write from 6502 microprocessor.
J4-10	$\overline{SEL\ 2}$	4K byte page address select for memory locations 2ØØØ-2FFF.
J4-11	$\overline{SEL\ 3}$	4K byte page address select for memory locations 3ØØØ-3FFF.
J4-12	$\overline{SEL\ 4}$	4K byte page address select for memory locations 4ØØØ-4FFF.
J4-13	$\overline{SEL\ 5}$	4K byte page address select for memory locations 5ØØØ-5FFF.
J4-14	$\overline{SEL\ 6}$	4K byte page address select for memory locations 6ØØØ-6FFF.
J4-15	$\overline{SEL\ 7}$	4K byte page address select for memory locations 7ØØØ-7FFF.
J4-16	$\overline{SEL\ 8}$	4K byte page address select for memory locations 8ØØØ-8FFF.
J4-17	$\overline{SEL\ 9}$	4K byte page address select for memory locations 9ØØØ-9FFF.
J4-18	$\overline{SEL\ A}$	4K byte page address select for memory locations AØØØ-AFFF.
J4-19	$\overline{SEL\ B}$	4K byte page address select for memory locations BØØØ-Bfff.
J4-22	$\overline{RES}$	Reset for 6502 microprocessor. Note: connected to 74LS00 output.
J4-23	$\overline{RDY}$	Ready line to the microprocessor.
J4-24	$\overline{NMI}$	Non maskable interrupt to microprocessor.
J9-1	GND	Logic ground.
J4-2	BDØ	Data bit Ø. Buffered.
J4-3	BD1	Data bit 1. Buffered.
J4-4	BD2	Data bit 2. Buffered.
J4-5	BD3	Data bit 3. Buffered.
J4-6	BD4	Data bit 4. Buffered.
J4-7	BD5	Data bit 5. Buffered.
J4-8	BD6	Data bit 6. Buffered.
J4-9	BD7	Data bit 7. Buffered.
J4-20	$\overline{RAS}$	Dynamic RAM RAS.
J4-21	$\overline{CAS}$	Dynamic RAM CAS.
J4-25	GND	Logic Ground.

Note that the 40 top edge "B" connections (or pins) are ground returns for the corresponding 40 lower edge "A" connections.

ADDRESSING MODES

ACCUMULATOR ADDRESSING - This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

IMMEDIATE ADDRESSING - In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

ABSOLUTE ADDRESSING - In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

ZERO PAGE ADDRESSING - The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

INDEXED ZERO PAGE ADDRESSING - (X, Y indexing) - This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

INDEXED ABSOLUTE ADDRESSING - (X, Y indexing) - This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes on the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

IMPLIED ADDRESSING - In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

RELATIVE ADDRESSING - Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

INDEXED INDIRECT ADDRESSING - In indexed indirect addressing (referred to as (Indirect,X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of the addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

INDIRECT INDEXED ADDRESSING - In indirect indexed addressing (referred to as (Indirect),Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

ABSOLUTE INDIRECT - The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address which is loaded into the sixteen bits of the program counter.

Hexadecimal Conversion Table

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

	IMM 2	ZPAG 2	Z,X (I,X) 2	(I),Y 2	ABS 3	A,X 3	A,Y 3
ORA	09	05	01	11	0D	1D	19
AND	29	25	21	31	2D	3D	39
EOR	49	45	41	51	4D	5D	59
ADC	69	65	61	71	6D	7D	79
STA		85	81	91	8D	9D	99
LDA	A9	A5	A1	B1	AD	BD	B9
CMP	C9	C5	C1	D1	CD	DD	D9
SBC	E9	E5	E1	F1	ED	FD	F9

Op Code ends in -1, -5, -9, or -D

	IMM 2	ZPAG 2	Z,X 2	ABS 3	A,X 3	A,Y 3
ASL		06	16	0E	1E	
ROL		26	36	2E	3E	
LSR		46	56	4E	5E	
ROR		66	76	6E	7E	
STX		86		8E		96
LDX	A2	A6		AE		B6
DEC		C6	D6	CE	DE	
INC		E6	F6	EE	FE	

Op Code ends in -2, -6, or -E

	BPL 10	BMI 30
BVC	50	BVS 70
BCC	90	BCS B0
BNE	D0	BEQ F0

Branches -0

	ABS (IND)
JSR	20
JMP	4C 6C

Jumps

	IMM 2	ZPAG 2	Z,X 2	ABS 3	A,X 3
BIT		24		2C	
STY		84	94	8C	
LDY	A0	A4	B4	AC	BC
CPY	C0	C4		CC	
CPX	E0	E4		EC	

Misc. -0, -4, -C

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	BRK					RTI			RTS							
-8	PHP	CLC	PLP	SEC	PHA	CLI	PLA	SEI	DEY	TYA	TAY	CLV	INY	CLD	INX	SED
-A	ASL-A		ROL-A		LSR-A		ROR-A		TXA	TXS	TAX	TSX	DEX		NOP	

Single-byte Op Codes -0, -8, -A

MCS6501-MCS6505 MICROPROCESSOR INSTRUCTION SET – ALPHABETIC SEQUENCE

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-Or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift Right One Bit (Memory or Accumulator)
NOP	No Operation
ORA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

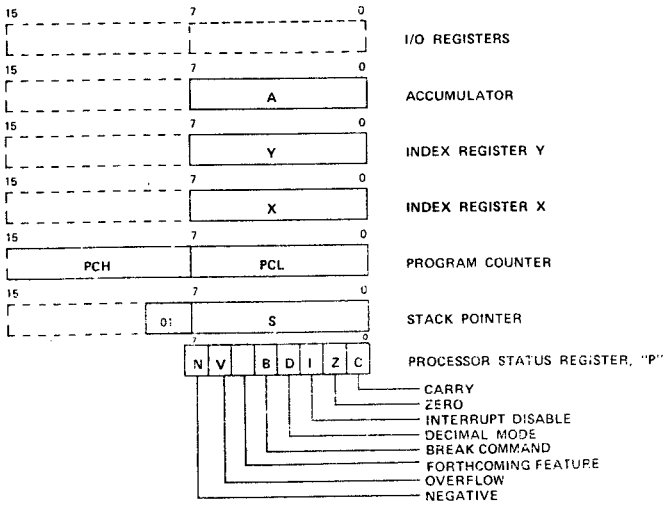
INSTRUCTION ADDRESSING MODES AND RELATED EXECUTION TIMES (in clock cycles)

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
AND	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
ASL	2	.	5	6	.	6	7	.	.	.	.	.	.
BCC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BCS	.	.	.	.	.	.	.	.	.	2**	.	.	.
BEQ	.	.	.	.	.	.	.	.	.	2**	.	.	.
BIT	.	.	3	.	.	4	.	.	.	.	.	.	.
BMI	.	.	.	.	.	.	.	.	.	2**	.	.	.
BNE	.	.	.	.	.	.	.	.	.	2**	.	.	.
BPL	.	.	.	.	.	.	.	.	.	2**	.	.	.
BRK	.	.	.	.	.	.	.	.	.	.	.	.	.
BVC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BVS	.	.	.	.	.	.	.	.	.	2**	.	.	.
CLC	.	.	.	.	.	.	.	.	2	.	.	.	.
CLD	.	.	.	.	.	.	.	.	2	.	.	.	.
CLI	.	.	.	.	.	.	.	.	2	.	.	.	.
CLV	.	.	.	.	.	.	.	.	2	.	.	.	.
CMP	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
CPX	.	2	3	.	.	4	.	.	.	.	.	.	.
CPY	.	2	3	.	.	4	.	.	.	.	.	.	.
DEC	.	.	5	6	.	6	7	.	.	.	.	.	.
DEX	.	.	.	.	.	.	.	.	2	.	.	.	.
DEY	.	.	.	.	.	.	.	.	2	.	.	.	.
EOR	.	2	3	4	.	4	4*	4*	.	.	6	5	.
INC	.	.	5	6	.	6	7	.	.	.	.	.	.
INX	.	.	.	.	.	.	.	.	2	.	.	.	.
INY	.	.	.	.	.	.	.	.	2	.	.	.	.
JMP	.	.	.	.	.	3	.	.	.	.	.	.	5

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
JSR	.	.	.	.	.	6	.	.	.	.	.	.	.
LDA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
LDX	.	2	3	.	4	4	.	4*	.	.	.	.	.
LDY	.	2	3	4	.	4	4*	.	.	.	.	.	.
LSR	2	.	5	6	.	6	7	.	.	.	.	.	.
NOP	.	.	.	.	.	.	.	.	2	.	.	.	.
ORA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
PHA	.	.	.	.	.	.	.	.	3	.	.	.	.
PHP	.	.	.	.	.	.	.	.	3	.	.	.	.
PLA	.	.	.	.	.	.	.	.	4	.	.	.	.
PLP	.	.	.	.	.	.	.	.	4	.	.	.	.
ROL	2	.	5	6	.	6	7	.	.	.	.	.	.
ROR	2	.	5	6	.	6	7	.	.	.	.	.	.
RTI	.	.	.	.	.	.	.	.	6	.	.	.	.
RTS	.	.	.	.	.	.	.	.	6	.	.	.	.
SBC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
SEC	.	.	.	.	.	.	.	.	2	.	.	.	.
SED	.	.	.	.	.	.	.	.	2	.	.	.	.
SEI	.	.	.	.	.	.	.	.	2	.	.	.	.
STA	.	.	3	4	.	4	5	5	.	.	6	6	.
STX*	.	.	3	.	4	4	.	.	.	.	.	.	.
STY**	.	.	3	4	.	4	.	.	.	.	.	.	.
TAX	.	.	.	.	.	.	.	.	2	.	.	.	.
TAY	.	.	.	.	.	.	.	.	2	.	.	.	.
TSX	.	.	.	.	.	.	.	.	2	.	.	.	.
TXA	.	.	.	.	.	.	.	.	2	.	.	.	.
TXS	.	.	.	.	.	.	.	.	2	.	.	.	.
TYA	.	.	.	.	.	.	.	.	2	.	.	.	.

* Add one cycle if indexing across page boundary
** Add one cycle if branch is taken, Add one additional if branching operation crosses page boundary

PROGRAMMING MODEL MCS650X



* Solid line indicates currently available features
Dashed line indicates forthcoming members of family

Table: Code assignments for "Command Mode" of operation.

(SENT AND RECEIVED WITH ATN TRUE)

b7 / b6 / b5 / Bits ②					0 0 0	① MSG	0 0 1	MSG	0 1 0	MSG	0 1 1	MSG	1 0 0	MSG	1 0 1	MSG	1 1 0	MSG	1 1 1	MSG
b4 ↓	b3 ↓	b2 ↓	b1 ↓	COLUMN → ROW ↓	0		1		2		3		4		5		6		7	
0	0	0	0	0	NUL		DLE		SP	MLA ASSIGNED TO DEVICE	0	MLA ASSIGNED TO DEVICE	@	MTA ASSIGNED TO DEVICE	P	MTA ASSIGNED TO DEVICE	`	MEANING DEFINED BY PCG CODE	p	MEANING DEFINED BY PCG CODE
0	0	0	1	1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
0	0	1	0	2	STX		DC2		"		2		B		R		b		r	
0	0	1	1	3	ETX		DC3		#		3		C		S		c		s	
0	1	0	0	4	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
0	1	0	1	5	ENQ	PPC ③	NAK	PPU	%		5		E		U		e		u	
0	1	1	0	6	ACK		SYN		&		6		F		V		f		v	
0	1	1	1	7	BEL		ETB		'		7		G		W		g		w	
1	0	0	0	8	BS	GET	CAN	SPE	(		8		H		X		h		x	
1	0	0	1	9	HT	TCT	EM	SPD	)		9		I		Y		i		y	
1	0	1	0	10	LF		SUB		*		:		J		Z		j		z	
1	0	1	1	11	VT		ESC		+		;		K		[		k		{	
1	1	0	0	12	FF		FS		,		<		L		\		l		|	
1	1	0	1	13	CR		GS		-		=		M		]		m		}	
1	1	1	0	14	SO		RS		.		>		N		^		n		~	
1	1	1	1	15	SI		US		/		?		O		—		o		DEL	

④

ADDRESSED COMMAND GROUP (ACG) — UNIVERSAL COMMAND GROUP (UCG) — LISTEN ADDRESS GROUP (LAG) — TALK ADDRESS GROUP (TAG)

PRIMARY COMMAND GROUP (PCG) — SECONDARY COMMAND GROUP (SCG)

NOTES:
① MSG = INTERFACE MESSAGE
② b_1 = DI01 . . . b_7 = DI07
③ REQUIRES SECONDARY COMMAND
④ DENSE SUBSET (COLUMN 2 THROUGH 5). ALL CHARACTERS USED IN BOTH COMMAND & DATA MODES.

Index: Transactors 1 - 6, Vol 2.

Transactor #1

Watching a Cassette Load.....J. Butterfield
Decimal/Hex Conversion.....J. Butterfield
The LIST Chain.....Karl J. Hildon
16/32 K PET Notes.....Jim Russo
Book Review: BASIC for Home Computers.....J. Butterfield

Transactor #2

PRINT Speed-Up.....Karl J. Hildon
Bits and Pieces
Merging PET Programs.....J. Butterfield
PET to Teletype Interface.....Lt. W. Hawes
Additional I/O Interface.....Kevin Erler
RAM Memory Map for Upgrade ROM.....J. Butterfield

Transactor #3

FREE Poster Coupon
The Append Wedge.....Bill Seiler
PET Variable Storage.....Karl J. Hildon
Machine Language IEEE Bus Handshake.....John A. Cooke
ROM Memory Map for Upgrade ROM.....J. Butterfield

Transactor #4

Computed GOTO.....Brad Templeton
Indenting Program Text.....Bill Seiler
More PET "Quirks".....Rick Ellis
Ifless Decisions.....Karl J. Hildon
Disabling The STOP Key.....J. Butterfield
A Fast Sort.....J. Butterfield
Centronics P1 Modification.....Rick Hoffman
Commodore International Offices

Transactor #5

Case Converter
Bits and Pieces
Memory Map: Original to Upgrade ROM.....J. Butterfield
Memory Expansion: Cost $0.00.....Karl J. Hildon
Casstte File END Markers.....J. Butterfield
Attention Multi-Peripheral Users
?LOAD Error.....F. VanDuinen
Screen Print Routine.....J. Butterfield
The IEEE-488 Bus.....J. Butterfield
D.R.I.P.....F. VanDuinen
FOR/NEXT Loop Structures.....J. Butterfield

Transactor #6

Inside The 2040 Disk Drive.....J. Butterfield
Memory Organization Table
Printer Formatting.....K. Lantz
Bits and Pieces
Screen POKE Table.....CBM Japan
Screen I/O.....Karl J. Hildon
ROM Entry Points.....J. Butterfield
Infinitely Long PET Programs.....Henry Troupe
DOS Support.....R.J. Fairbairn
Random Access File Indexing.....Jim Hindson
Indexing Programs On Cassette.....M. Casey
Micros in Archaeological Fieldwork.....G.D. Hathaway

Product Announcements

PET Reset Switch.....Transactor #1 & #2
MTU Visible Memory..... "" #1
TTY Interface..... "" #2
Memory Expansion..... "" #5
COMPUTE Magazine..... "" #5